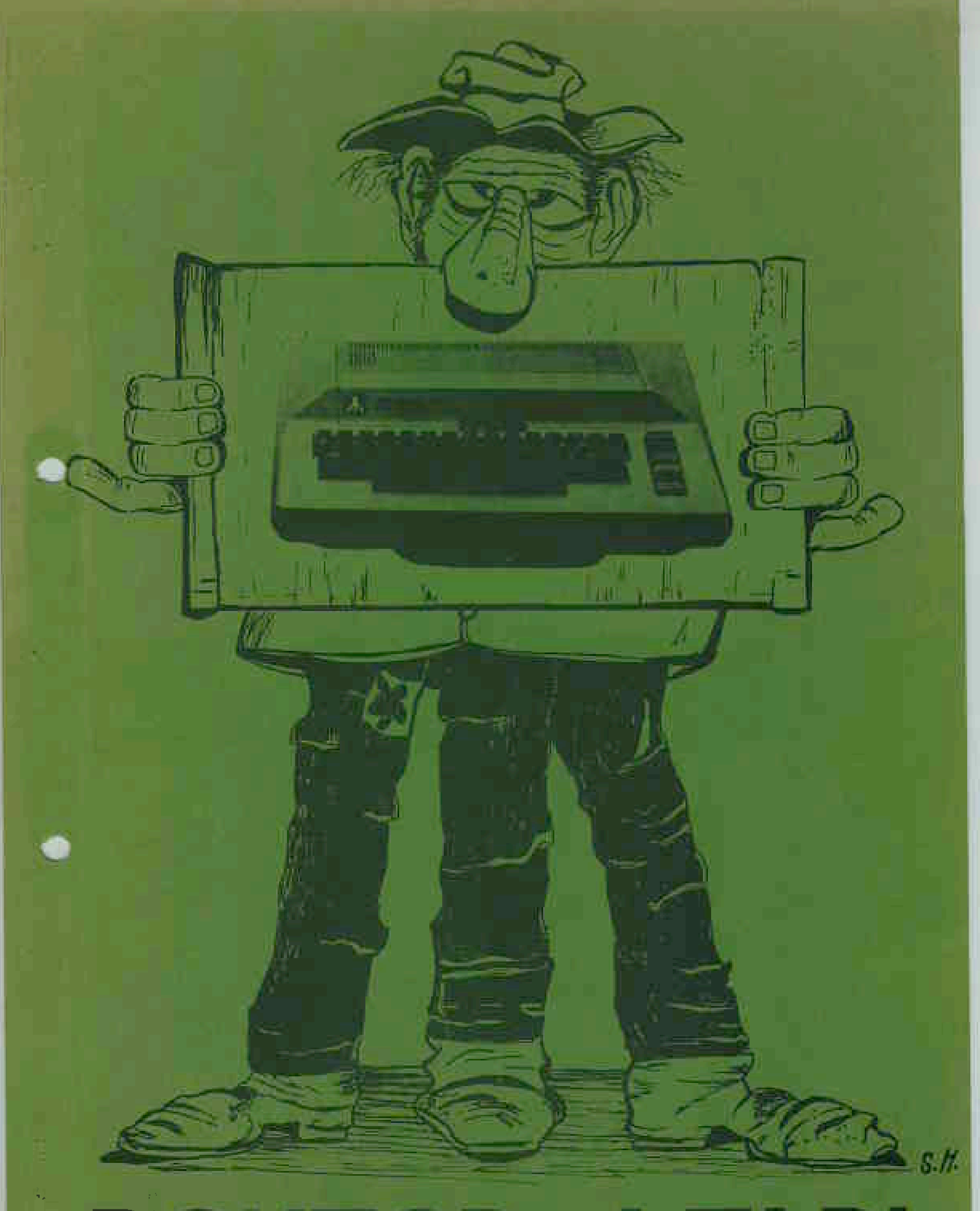


DOKTOR A.TARI

PAGE FLIPPING (SEITENWAHL)



Die zusätzliche Kenntnis der Technik des "PAGE FLIPPING" ist sehr zu empfehlen, da sie oft angewand werden kann.

Worin besteht dieses Wählen von Bildseiten?

Es ist eine für kleine Computer eigentlich ungewöhnliche Methode. Sie kann auf der ATARI angewand werden, weil es auf dieser Maschine möglich ist, sowohl die Displayliste als auch die zu projezierende Daten an beliebiger Stelle im Speicher unterzubringen.

Eine Displayliste ist eine kleine Reihe von Instruktionen, die der Computer braucht, um zu wissen, wie die Bilddaten auf den Monitor zu bringen sind. Auf dem ATARI kann man nun die Information für zahlreiche Bildmuster (Bildseiten), gleich ob nun Text oder Graphik, vorspeichern, um sie an späterer Stelle im Programm direkt abzurufen.

Zuerst einige Grundlagen:
Das Maschinengedächtnis (RAM) läßt sich sowohl zum Einspeichern wie zum Abrufen beliebig adressieren. Man spricht von RAM gewöhnlich in Einheiten von Bytes. Ein Speicher kann z.B. 16 K , 24 K, 32 K, 40 K oder 48 K Bytes besitzen. Ein Block von einem K enthält genau gesagt 1024 Bytes. Um Bildmuster zu überspringen, bewegt man die Speicheradressierung in Strecken von jeweils 256 Bytes weiter, diese Strecken nennt man auch "SEITEN". 4 Seiten ergeben 4 mal 256 Bytes, also 1024 Bytes = 1 K, wobei K für "KILO", also 1000 steht. Der Ausdruck "Seitenwahl" ist in sofern unkorrekt, als eigentlich nicht um jeweils 256 Bytes weiter gesprungen wird. Es handelt sich vielmehr um das Anwählen einzelner Bilddatensätze.

Unsere Beispiele zeigen 2 Methoden des Page Flipping. Man sollte diese Beispiele möglichst intensiv modifizieren und für eigene Zwecke einsetzen, um die Flipping-Technik zu üben.

Bei Speichern von 32 K und mehr sollte man auf hohe Auflösung , Graphikgang 8, umprogrammieren und sowohl Text wie Graphik einbauen.

1. Methode:

Beim Ansetzen eines Graphikgangs, 0-8,bereitet der Computer normalerweise eine Speicherzone für die Displayliste und eine für die Bilddaten direkt unterhalb des oberen Speicherrandes vor. Die 1. Methode von der hier die Rede ist, sagt dem Computer nach der 1. Bildprojektion: "Die Speicherinformation steht nicht an der gewohnten Stelle, sondern weiter unten. Daher muß eine zusätzliche Displayliste und eine zusätzliche Datenzone weiter unten eingerichtet werden !"

In dem man dies so oft wie nötig und möglich tut, hat eine Reihe fertig vorbereiteter Bildmuster und Displaylisten im Speicher, deren Anfangsadressen jeweils per Programm direkt angewählt werden können.

Beispielprogramm 1:

Man sieht bei Anwendung dieses Beispielsprogramms, das es einige Zufallszeilen zeichnet, die dann verschwinden und durch neue Zeilen ersetzt werden. Beide Bildmuster scheinen abwechselnd aufzutauchen und wieder zu verschwinden.

Die Programmzeilen 10-25 zeichnen ein Bild wie bei gewöhnlichen COLOR-, PLOT- und PRINT-Befehlen. Vergl. hierzu das BASIC - Handbuch.

Zeile 30 speichert den ursprünglichen Speicheranfang (als Anzahl von 256-Byte - Abschnitten), der vom Register 106 durch PEEK-Befehl abgerufen wurde. Zeile 40 speichert die beiden Zahlen, die die Lage der DL bezeichnen. Um alle Speicherplätze adressieren zu können, braucht man 2 Zahlen a 8 Bit. Der untere Teil davon adressiert von 0-255, der obere Teil von 0 - 255 mal 256 (vergl. Handbuch). Hier wird nur der Register 561 stehende Adressteil benutzt, da wir im Speicher um jeweils ganze Seiten hinunterrücken wollen und in 561 die jeweiligen Seitennummern gespeichert sind.

Der untere Teil der Adressierung wurde von uns trotzdem mit in das Programm genommen, damit jeder nach belieben auch mit diesem experimentieren kann.

Zeile 50 besagt: "Das obere Speicherende ist um 32 Seiten (8K Bytes) gegenüber dem ursprünglichen Wert nach unten verschoben."

Das ist mehr als in diesem Beispiel nötig wäre. Der zusätzliche Speicherraum ermöglicht es, später zusätzliche Bildmuster einzufügen. Zeile 60-85 zeichnen einige Graphiken in der üblichen Weise. Der Computer fragt jetzt Register 106 ab und stellt den von uns dorthin gebrachten Wert fest. Er wird hier getäuscht, indem wir ihn veranlasen, die neue DL und die zugehörigen Bilddaten um 32 "SEITEN" niedriger zu plazieren als es normalerweise geschehen wäre. Es ist klar, daß die Anzahl von Seiten, um die man weiterrückt, dem für den jeweiligen Graphikgang benötigten Speicherraum entsprechen muß. Den entsprechenden Wert ersieht man aus dem Graphikteil des BASIC-Handbuches.

In unserem Fall würde Gang 6 eingesetzt, sodaß 2K bzw. 8 Seiten genug gewesen wären. Wir ließen jedoch noch eine Reserve, damit das Programm leichter kopierbar und ergänzbar ist.

Zeile 70 speichert die Lageadresse dieser 2. DL. Das kann vor den Graphikstatements gemacht werden (wie in unserem Fall) oder auch danach.

Zeile 90 führt das eigentliche Flipping durch , indem sie in 561 den oberen Teil der Adresse bringt (POKE), bei der die 1. Displayliste beginnt. Diese Adresse wird dann durch die Adresse der 2. DL ersetzt, diese wieder durch die der 1. DL und so weiter.

Anmerkung :
Durch Programmzeile 110 in Beispiel 1 wird Register 106 wieder auf den ursprünglichen Wert gebracht. Das ist notwendig, wenn man die oben beschreibene Täuschungstechnik im Programm fortgesetzt anwendet, ohne das zwischdurch die Maschine angeschaltet wird. Schaltet man jedoch ab, so wird Register 106 beim nächsten Einschalten autumatisch richtig gestellt.

Weiter ist zu beachten, daß in diesem Programm nicht RESET oder BREAKE gedrückt werden darf, es sei denn, eine Panne liegt vor. Immer wenn ein Beispielprogramm sich festgelaufen hat, RESET drücken, RUN eintasten und RETURN drücken !

Beispielprogramm 2:

Dies Programm ist aus dem 1. entwickelt und zeigt, wie man die Grundidee des Bildwechselns erweitern kann. Wir richten in diesem Fall 4 Displaylisten ein und printen 4 einfache Mitteilungen auf den Monitor. Sobald ein Knopf gedrückt wird, erscheint die Mitteilung sofort im Bild. Es wird dem Benutzer hier wohl klar, daß zwischen dem Flippen zwischen 4 einzeiligen Bildtexten und dem zwischen 4 vollständig aufgefüllten Monitormustern kein prinzipieller Unterschied besteht.

In beiden Fällen wird bei Knopfdruck jeweils ein anderer Speicherbereich projeziert. Man kann nun zur Übung mehr und mehr Zeilen einfügen und aufs Bild bringen, indem man einfach nur Werte zwischen die Anführungsstriche auf den Zeilen 20, 70, 110 und 150 setzt.

Die Programmzeilen 10-40 schreiben Text und speichern die benötigten Werte für die Displayliste 1 (DL1).

Auf 50-80 wird Register 106 um 8 Seiten (2K) nach unten verändert, außerdem mehr Text geschrieben, und diese neuen Werte werden gespeichert.

Die Zeilen 90-120 wiederholen dies für den Bereich, der weitere 8 Seiten unter dem vorigem liegt. Der Text ist ebenfalls anders. Entsprechendes geschieht bei 130-160 für einen 4. Bildablauf.

In den Zeilen 168-185 ist die Bedienung von Taste 1-4 (oder von entsprechenden Tasten) vorgesehen, sodaß die Codes auf 190 - 220 dem Computer angeben können, welcher der vorbereiteten Displaylisten gewählt wird.

Der Test CH=12 dient zur Feststellung, ob ein Programmsprung durch RETURN vorliegt oder nicht. Der POKE auf 255 an 764 löscht das Register, in welchem der interne Code für die zuletzt gedrückte Taste gespeichert ist. Die Variable CH auf Programmzeile 170 gibt dem System an, welche Taste soeben gedrückt wurde.

Anmerkung:
Man kann in dieses Programm eigene Texte einfügen oder die Maschine Text von der Tastatur bzw. von Diskette lesen lassen und die ersten 960 Bytes (entsprechend dem Umfang von Graphik 0) auf Seite 1 plazieren, die nächsten 960 auf Seite 2 u.s.w. .
Wie man Daten von der Tastatur oder von der Diskette einliest, wird in den Handbüchern beschrieben.

Beispielprogramm 3:

Beispiel 3 entspricht dem vorigem Beispiel, nur daß hier Graphik statt Text verarbeitet wird. Man braucht für die Textbefehle lediglich Graphikbefehle einzusetzen. Wir haben hier einfache Balkengraphen programmiert, doch können natürlich auch komplizierte Bilder gezeichnet werden, denn das Projektionsmaterial selbst hat mit der Methode des Bildaufbaus nichts zu tun. Auch wenn man den ganzen Monitor ausfüllt, fragt der Computer immer nur die Displayliste ab, um die Herkunftsadressen für die Daten zu bekommen.

Das zeitraubende Überprüfen der Projektion von komplizierten Bildern in jedem Einzelfall kann man sich jetzt sparen. Man speichert sie abrufbereit komplett in der Maschine.

Beispielprogramm 4:

Dies Beispiel zeichnet auf jeder Seite einen bestimmten Umriß. Durch abwechselnde Anwahl der Bildseiten kann man diese Umrisse in Bewegung bringen. Ein solches Programm läßt sich zweckmäßigerweise auf Platte oder Kassette aufheben und nach Bedarf mit irgentwelchen Umrissen ergänzen. Auch könnte man eine Geschäftsplanung in Verbindung mit einer Umsatzkurve damit ablaufen lassen. Andere Anwendungen sind leicht denkbar.

Beispielprogramm 5:

Man beachte hier die Zeile 7, auf der der Wert von Register 559 festgehalten wird. Zeile 13 bringt dann eine 0 in das Register 559. Auf diese Weise wird das Bild abgeschaltet, sodaß Bilder entstehen aber nicht sichtbar werden. Der Computer wird bei dieser Programmierung um etwa 30% schneller, je nach Graphikgang. Register 559 steuert nähmlich den ANTIC-Chip, der das Bildmaterial projeziert. Zum Anschalten des ANTIC-Chips setzt man den Originalwert (gespeichert in Register "NON") , zum Abschalten setzt man die 0. Letztere geschieht auf Zeile 169. Das funktioniert bei jedem beliebigen Programm!

Methode 2:
==========



Diese Methode unterscheidet sich nur leicht von der ersten, bietet aber bessere Kontrollmöglichkeiten, wie das Beispiel zeigt:

Beispielprogramm 6:
===================

Statt einer Variablen "A" benutzen wir jetzt "PIE 106" zum Speichern der Original-Seitennummern. Das macht alles übersichtlicher. PIE 106 steht also für den Wert, der in 106 GEPOKT werden soll.

Auf Zeile 10 wird die Adresse des DL - Anfangs als eine Dezimalzahl gespeichert. Zeile 15 Speichert die Zahl, nach der wir suchen; sie erscheint 5 Bytes hinter dem Anfang der Displayliste , sodaß wir den PEEK auf DL+5 setzen, also auf die 6. Zeile in der DL.

Zeile 16 verkleinert das Register 106 um einige Seiten (in diesem Fall um 4), zugleich wird hier aber auch ein neuer benötigter Wert gespeichert: Der Inhalt von Register 89. Dies enthält eine Kopie des Wertes in DP+5, also der Angabe der Anfangsadresse der DL.

Der Computer fragt die Displayliste jedesmahl ab, wenn ein Graphics-Befehl erscheint , und speichert den vorgefundenen Wert hier ab, damit er jederzeit über die Startadresse informiert ist.

Nach erfolgtem Graphikabruf kann man diese Zahl in Register 89 ändern, um den Computer zu "Täuschen", damit er auf die von uns gewünschte Adresse umschaltet.

Nachdem die Werte in diesen beiden Registern weit genug nach unten gepokt sind, kann man etwas Text in den neuerreichten Speicherbereich schreiben (Zeile 17). Das kann wiederholt werden, solange der RAM ausreicht.

Während nun die Zeilen 30-38 die gewünschte Bildseite abrufen, ändern die Zeilen 40 oder 45 den Wert innerhalb der 1. DL, der angibt, woher diese ihr Datenmaterial nimmt. Eine 2.,3. oder 4. Liste , wie im 1. Beispiel, bauen wir hier nicht auf.

Das bedeutet also, man kann durch einfachen Wechsel eines Wertes am Beginn der 1. Diplayliste die verschiedensten Daten aus allen Speicherbereichen projezieren. Das könnte leicht über einen Spielhebel gesteuert werden. Auch könnte man die Absicht haben, ein schon fertiges Bild zu überprüfen, während man andere programmiert.

Beispielprogramm 7:

Dies Programm zeigt die oben erwähnten Möglichkeiten:

Man kann hier nicht nur 2 verschiedene Bilder betrachten (0 wählt Bild 1 an, 4 wählt Bild 2) , sondern durch Eingabe anderer positiver Ziffern die Speicherbereiche weiter unten sehen, bzw. durch Eingabe negativer Ziffern die oberen Bereiche bis zum Anfang. Was man auf dem Monitor sieht, sind die alphanumerischen equivalente des BASIC-Programms, der Bilddaten, des Operationssystems oder sonstiger Teile des Speichers.

Die Änderungspunkte in diesem Programm sind :

Zeilen 20 und 33	: Zahleneingabe,
Zeile 35	: fragt diese Zahl ab, ob sie nicht zu groß ist,
Zeile 40	: spricht die DL neu an, jedoch mit dem vom Benutzer eingegebenen Wert des zuvor festgelegten Wertes 4.

Beispielprogramm 8:

Das letzte Beispiel entspricht fast genau dem 7., nur das es mit Einsatz von Farbeffekten die Speicherinhalte projeziert. Man sollte in den beiden letzten Beispielen unbedingt auch negative Will-Ziffern eintasten. Die negativen wählen die oberen Speicherbereiche an, die positiven die unteren. Man kann mit diesem Beispiel alle erdenklichen Speicherzonen erfassen.

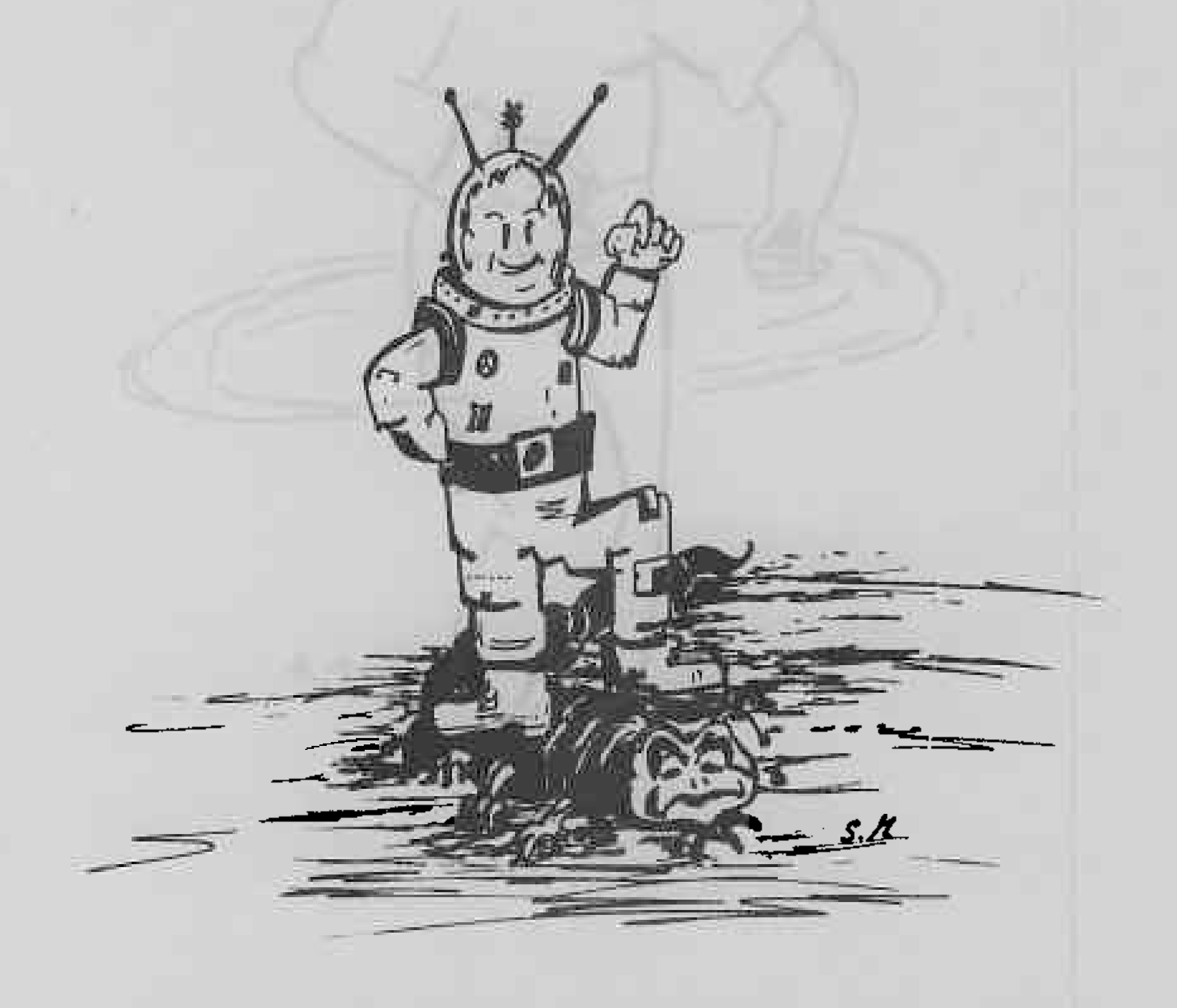
Soweit dieses Thema. Der Benutzer sollte alle unsere Programme zu modifizieren versuchen, sodaß er sowohl die Methodik versteht als auch die Programme selbst für eigene Zwecke verbessern kann.



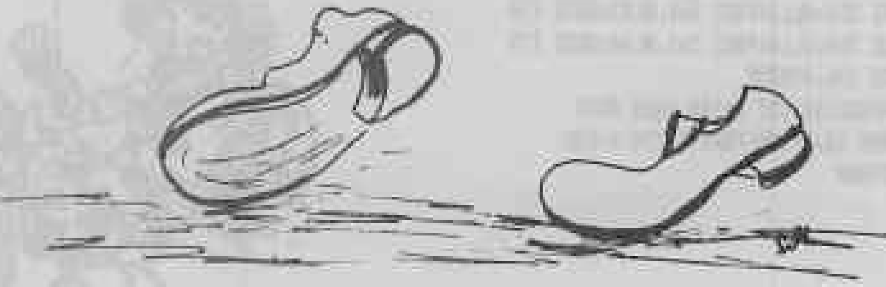
Die in der Übersicht nicht erklärten Gänge, dienen Spezialaufgaben wie Unterlängen bei Kleinbuchstaben oder mehrfarbiger Schrift. Sie werden in späteren Veröffentlichungen behandelt. Dasselbe gilt für die Graphikgänge 9-11, bei denen Prioritätsregister eingesetzt werden müssen.



```
1 REM BEISPIEL 1
2 REM
3 REM
4 REM
10 GRAPHICS 6
12 GOSUB 4000
15 TRAP 20
20 COLOR 1:FOR I=1 TO 20:COLOR 2*RND(4):DRAWTO 140*RND(4),70*RND(9):NEXT I
25 ? "THIS IS FLIPPING BETWEEN TWO AREAS OF MEMORY.  PRESS RETURN TO CONTINUE."
30 A=PEEK(106)
40 DLISTL1=PEEK(560):DLISTH1=PEEK(561)
50 POKE 106,A-32
60 GRAPHICS 6
61 GOSUB 4000
70 DLISTL2=PEEK(560):DLISTH2=PEEK(561)
75 TRAP 80
80 COLOR 1:FOR I=1 TO 20:COLOR 2*RND(4):DRAWTO 140*RND(4),70*RND(9):NEXT I
85 ? "THIS IS FLIPPING BETWEEN TWO AREAS OF MEMORY.  PRESS RETURN TO CONTINUE."
86 POKE 764,255
90 POKE 561,DLISTH1:FOR W=1 TO 2:NEXT W:POKE 561,DLISTH2:IF PEEK(764)=12 THEN 110
100 GOTO 90
110 POKE 106,A:STOP
4000 X=PEEK(16):IF X=128 THEN 4020
4010 POKE 16,X-128:POKE 53774,X-128
4020 RETURN
```



```
1 REM BEISPIEL 2
2 REM
3 REM
4 REM
5 TRAP 10
10 GRAPHICS 0
20 POSITION 4,10:? "THIS IS PAGE 1.":POSITION 2,20:? "PRESS 1,2,3 OR 4 FOR THAT PAGE."
25 ? "PRESS RETURN TO CONTINUE."
30 A=PEEK(106)
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 0
70 POSITION 10,10:? "THIS IS PAGE 2.":POSITION 2,20:? "PRESS 1,2,3 OR 4 FOR THAT PAGE."
75 ? "PRESS RETURN TO CONTINUE."
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
90 POKE 106,A-16
100 GRAPHICS 0
110 POSITION 15,10:? "THIS IS PAGE 3.":POSITION 2,20:? "PRESS 1,2,3 OR 4 FOR THAT PAGE."
115 ? "PRESS RETURN TO CONTINUE."
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ******************************
130 POKE 106,A-24
140 GRAPHICS 0
150 POSITION 20,10:? "THIS IS PAGE 4.":POSITION 2,20:? "PRESS 1,2,3 OR 4 FOR THAT PAGE."
155 ? "PRESS RETURN TO CONTINUE."
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ******************************
168 POKE 764,255:GOSUB 4000
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:STOP
4000 X=PEEK(16):IF X=128 THEN 4020
4010 POKE 16,X-128:POKE 53774,X-128
4020 RETURN
```



```
1 REM BEISPIEL 3
2 REM
3 REM
4 REM
5 TRAP 10
10 GRAPHICS 5
15 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
20 COLOR 2:PLOT 10,34:DRAWTO 10,25:DRAWTO 5,25:POSITION 5,34:POKE 765,2:XIO 18,#6,0,0,"S:"
25 ? "PRESS 1,2,3, OR 4 FOR BAR GRAPHS OF":? "THE YEARS 1981,1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
30 A=PEEK(106)
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 5
65 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
70 COLOR 3:PLOT 15,34:DRAWTO 15,20:DRAWTO 10,20:POSITION 10,34:POKE 765,3:XIO 18,#6,0,0,"S:"
75 ? "PRESS 1,2,3, OR 4 FOR BAR GRAPHS OF":? "THE YEARS 1981,1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
90 POKE 106,A-16
100 GRAPHICS 5
105 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
110 COLOR 1:PLOT 20,34:DRAWTO 20,15:DRAWTO 15,15:POSITION 15,34:POKE 765,1:XIO 18,#6,0,0,"S:"
115 ? "PRESS 1,2,3, OR 4 FOR BAR GRAFHS OF":? "THE YEARS 1981,1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ******************************
130 POKE 106,A-24
140 GRAPHICS 5
145 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
150 COLOR 2:PLOT 25,34:DRAWTO 25,10:DRAWTO 20,10:POSITION 20,34:POKE 765,2:XIO 18,#6,0,0,"S:"
155 ? "PRESS 1,2,3, OR 4 FOR BAR GRAFHS OF":? "THE YEARS 1981,1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ******************************
168 POKE 764,255:GOSUB 4000
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:STOP
4000 X=PEEK(16):IF X=128 THEN 4020
4010 POKE 16,X-128:POKE 53774,X-128
4020 RETURN
```

```
1 REM BEISPIEL 4
2 REM
3 GRAPHICS 0
5 TRAP 10
10 GRAPHICS 5
15 COLOR 1
20 READ X,Y:IF X=0 THEN 30
25 PLOT X,Y:GOTO 20
27 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
28 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
29 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
30 A=PEEK(106)
35 ? "     HUP!"
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 5
6[illegible]OLOR 1:RESTORE 77
65 ? "     TWO!"
70 READ X,Y:IF X=0 THEN 80
75 PLOT X,Y:GOTO 70
77 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
78 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
79 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
90 POKE 106,A-16
100 GRAPHICS 5
103 ? "     THREE!"
105 COLOR 1:RESTORE 117
110 READ X,Y:IF X=0 THEN 120
115 PLOT X+20,Y:GOTO 110
117 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
118 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
119 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
120 DLL3=PEEK(560):DLH3=PEEK(561)
1[illegible] REM ******************************
130 POKE 106,A-24
140 GRAPHICS 5
143 ? "     FOUR!"
145 COLOR 1:RESTORE 157
150 READ X,Y:IF X=0 THEN 160
155 PLOT X+20,Y:GOTO 150
157 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
158 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
159 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ******************************
168 POKE 764,255
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:STOP
```



```
1 REM BEISPIEL 5
2 REM
3 REM
4 REM
5 TRAP 10
7 NON=PEEK(559)
10 GRAPHICS 5
13 POKE 559,0
15 COLOR 1
20 READ X,Y:IF X=0 THEN 30
25 PLOT X,Y:GOTO 20
27 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
28 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
29 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
30 A=PEEK(106)
35 ? "     HUP!"
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 5
62 POKE 559,0
63 COLOR 1:RESTORE 77
65 ? "     TWO!"
70 READ X,Y:IF X=0 THEN 80
75 PLOT X,Y:GOTO 70
77 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
78 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
79 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
90 POKE 106,A-16
100 GRAPHICS 5
102 POKE 559,0
103 ? "     THREE!"
105 COLOR 1:RESTORE 117
110 READ X,Y:IF X=0 THEN 120
115 PLOT X+20,Y:GOTO 110
117 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
118 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
119 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ******************************
```

```
130 POKE 106,A-24
140 GRAPHICS 5
142 POKE 559,0
143 ? "     FOUR!"
145 COLOR 1:RESTORE 157
150 READ X,Y:IF X=0 THEN 160
155 PLOT X+20,Y:GOTO 150
157 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
158 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
159 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ******************************
168 POKE 764,255
169 POKE 559,NON
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:STOP
```

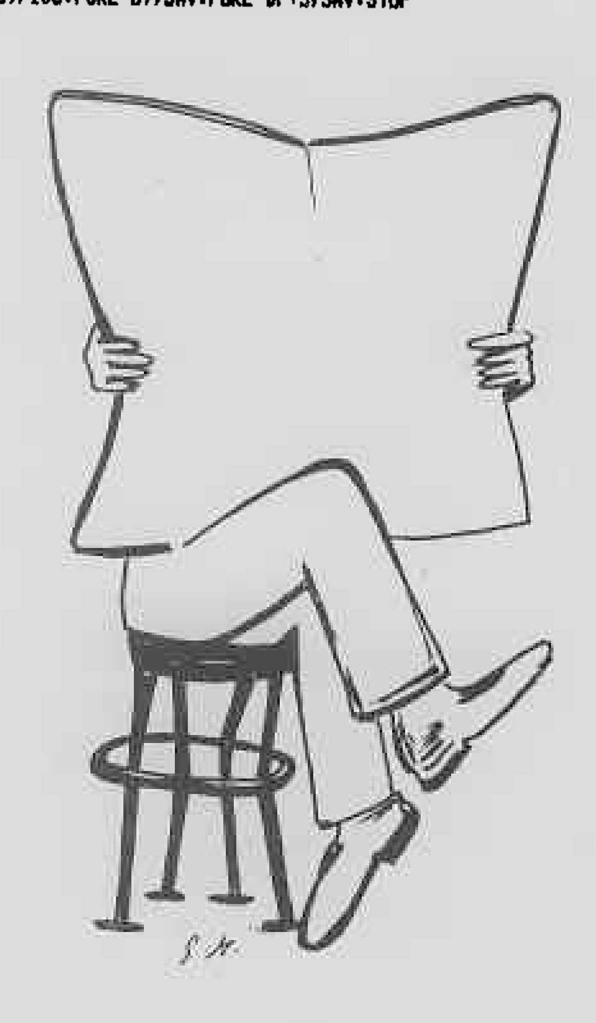
```
1 REM BEISPIEL 6
2 REM
3 REM
4 P106=PEEK(106)
5 ? "}":? "AT PAGE ONE!":? "PRESS 1 OR 2 FOR THAT PAGE."
7 ? "PRESS /ʃ↑1/$ TO GO ON."
10 DP=PEEK(560)+PEEK(561)*256
12 POKE 16,64
15 SAV=PEEK(DP+5)
16 POKE 106,P106-4:POKE 89,SAV-4
17 ? "AT PAGE TWO!":? "PRESS 1 OR 2 FOR THAT PAGE.":? "PRESS /ʃ↑1/$ TO GO ON."
30 POKE 764,255:TRAP 80
33 CH=PEEK(764)
35 IF CH=31 THEN 45
36 IF CH=12 THEN 60
37 IF CH=30 THEN 40
38 GOTO 33
40 POKE DP+5,SAV-4
43 GOTO 33
45 POKE DP+5,SAV
55 GOTO 33
60 POKE 106,P106:STOP
```

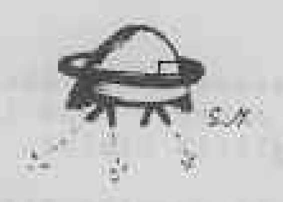


```
1 REM BEISPIEL 7
2 REM
3 REM
4 P106=PEEK(106)
5 ? "}":? "AT PAGE ONE!":? "PRESS ɛ↑ǂ/↑ AND ǂ↑↑ɿǂ# AT THE SAME    TIME TO GO ON.":? "PRESS BETWEEN 0 AND ";P106-5;
7 ? " TO LOOK AT    MEMORY IN 1/4 PAGE SCREEN INCREMENTS.":? "THEN PRESS RETURN. REPEAT AS DESIRED."
10 DP=PEEK(560)+PEEK(561)*256
12 POKE 16,64
15 SAV=PEEK(DP+5)
16 POKE 106,P106-4:POKE 89,SAV-4
17 ? "AT PAGE TWO!":? "   PRESS ɛ↑ǂ/↑ AND ǂ↑↑ɿǂ# AT THE SAME TIME TO GO ON"
20 DIM A(2)
25 POKE 53279,8
30 TRAP 30:Z=PEEK(53279):IF Z=6 THEN 80
33 INPUT A
35 IF A>(P106) THEN 60
40 POKE DP+5,SAV-A
55 GOTO 30
60 ? "NUMBER TOO LARGE, MUST BE LESS THAN";P106-5:? "WE ARE NOW AT 4 PAGES DOWN IN MEMORY"
65 POKE DP+5,SAV-4:POKE 89,SAV-4
70 GOTO 30
80 TRAP 40000:POKE 106,P106:POKE 89,SAV:POKE DP+5,SAV:STOP
```



```
1 REM BEISPIEL 8
2 REM
3 REM
4 P106=PEEK(106)
5 GRAPHICS 5:COLOR 1:PLOT 10,10:DRAWTO 10,20:DRAWTO 40,20:DRAWTO 40,10:DRAWTO 10,10
10 DP1=PEEK(560)+PEEK(561)*256
12 POKE 16,64
15 SAV1=PEEK(DP1+5)
16 POKE 106,P106-8
17 GRAPHICS 5:COLOR 2:PLOT 20,20:DRAWTO 20,30:DRAWTO 30,30:DRAWTO 30,20:DRAWTO 20,20
20 DIM A(2)
21 DP2=PEEK(560)+PEEK(561)*256
22 SAV2=PEEK(DP2+5)
25 POKE 53279,8
27 ? "PRESS ﾓﾔﾁﾒﾄ AND ﾒﾌﾔ1ﾒ8 TOGETHER TO GO ON."
30 TRAP 30:Z=PEEK(53279):IF Z=6 THEN 80
33 INPUT A
35 IF A)(P106) THEN 60
40 REM POKE DP1+5,SAV-A
41 POKE DP2+5,SAV1-A
55 GOTO 30
60 ? "NUMBER TOO LARGE, MUST BE LESS THAN";P106-5:? "WE ARE NOW AT 4 PAGES DOWN IN MEMORY."
65 POKE DP+5,SAV-4:POKE 89,SAV-4
70 GOTO 30
80 TRAP 40000:POKE 106,P106:POKE 89,SAV:POKE DP+5,SAV:STOP
```





DISPLAY LIST

Die Atarianlage besitzt einen Spezialchip (ANTIC), dessen Aufgabe es ist, Informationen vom Speicher auf den Bildschirm zu bringen. ANTIC ist eigentlich selbst ein Microprozessor und besitzt demgemäß einen eigenen Instruktionssatz.

Der Chip läßt sich freilich nicht, wie der Hauptprozessor, mit BASIC oder Assemblersprache programmieren, ist aber doch eine erhebliche Erweiterung der Leistungskapazität.

Die sogenannte DL (Display List) ist nun eine Informationsserie, die das ANTIC- System benutzt, um der Anlage zu sagen, was auf den Monitor gebracht werden soll und wie es zu projezieren ist.

Sie liefert also 1. die Graphik bzw. die Texte für das Bild, 2. die Adressen dieser Elemente.

Es gibt für diesen Bereich 4 Arten von Instruktionen, die mit Hilfe einiger BASIC-Statements in den Speicher gebracht werden.

Im Handbuch zur ATARI - Anlage werden (in Abb. 1) sogenannte OS-Arbeitsgänge, also Arbeitsgänge des Operationssystems, erwähnt. Die Instruktionen für die Display Listung erlauben den Einsatz dieser OS-Graphikgänge von 2-15. Dabei entspricht der normale Graphikgang NULL dem OS-Gang 2 und der normale Graphikgang 8 entspricht dem OS-Gang 15. Dazwischen liegen weitere BASIC-Gänge, die nicht ausdrücklich im Handbuch beschrieben werden, deren Gebrauch in dieser Information später erklärt werden sollen. Wenn im folgendem einfach von Graphikgängen die Rede ist, sind die normalen BASIC-Gänge gemeint, wenn es OS-Gang heißt, so ist ein Modus des Operatingsystems gemeint.

Weiter sind die Begriffe "Arbeitszeile" und "Pixelzeile" (Bildzeile) zu unterscheiden. Eine Pixelzeile ist die Reihe aus winzigen Punkten, die man dem Bildschirm sehen kann. Die Arbeitszeilen sind jeweils ein Strang aus mehreren solchen Bildzeilen, aus denen sich die Information für den Benutzer (z.B. eine Textzeile)zusammensetzt. So gibt die Tabelle für Graphikgang 0 in der Spalte "Pixelzeilen per Arbeitszeile" den Wert 8 an, sodaß also 8 Bildzeilen zusammen eine Zeile der tatsächlichen Information ergeben.

Diese Angaben braucht man, um Bilddarstellungen richtig programmieren zu können. Nimmt eine Arbeitszeile im Graphikgang 8 nur eine Bildzeile ein, während OS-Gang 5(ohne Entsprechung bei Normalgängen) 16 Bildzeilen umfaßt. Programmierung von OS-Gang 5 in der Displayliste führt das Bild also 16 mal so schnell auf wie Gang 8.

Beim Austausch von Gängen innerhalb eines DL-Programms sollte man darauf achten, daß letztenendes die Gesamtzahl der verbrauchten Zeilen gleich bleibt.

Wenn man zu wenige Bildzeilen vorsieht, "schrumpft" das Bild und der Rest des Monitors bleibt schwarz. Sieht man zu viele Zeilen vor, so fallen einige heraus und das Bild fängt unter Unständen an zu rollen. Dieser Vorgang kann dann mit der Vertikal-Kontrolle am Fernsehgerät nicht unterbunden werden.

Beispielprogramm 1:

Dieses Programm enthält alle Standard-Graphikgänge und ihre Displaylisten, einschließlich der GTI-Gänge 9,10 und 11. Zunächst soll Gang 0 eingegeben werden.

Die bei diesem Programm erscheinenden Zahlen sind die eines normalen Kleintextbildes.

Die Zahlen werden reihenweise gelesen. Die ersten drei Ziffern lauten 112. Sehen Sie bitte in der Tabbelle nach, aus der sich ergibt, das "112" die Maschine anweist, 8 Leerzeilen aufs Bild zu bringen. Drei solcher Befehle ergeben also 24 leere Zeilen oben im Bild. Es handelt sich um die Standardzahl von Leerzeilen, die zum Ausgleichen von individuellen Unterschieden zwischen den TV - Geräten notwendig sind (Overscan), damit wirklich alles wesentliche aufs Bild kommt.

Die nächste Zahl, eine 66, ist eine LMS- Instruktion. LMS=Load Memory Scan (Lade Speicherzeile). Das bedeutet: "Suche die Daten für die folgenden Arbeitsgänge, beginnend bei der in den nächsten beiden Zahlen gegebenen Adressen." Wir verwenden für die LMS-Zahl immer den Wert 64 + der OS-Gang-Zahl, in diesem Fall (Graphikgang 0) also 64 + 2 = 66. Graphikgang 0 entspricht nähmlich dem OS - Gang 2.

Die nächsten beiden Zahlen geben die Anfangsadresse der zu projezierenden Daten an. Um diesen Wert zu finden, nimmt man den zweiten Teil (oberes Byte) der Adresse, multipliziert ihn mit 256 und addiert den ersten Teil (unteres Byte) dazu. Wenn zum Beispiel der obere Teil 132 lauten würde und der untere 192, so würde der Beginn der Bilddaten bei 256 + 132 + 192 liegen, also 33984. Bringt man eine Information (Buchstabe, Zahl oder Graphik- Element) in diese Adresse, so erscheint sie oben links im Bild (in diesem Fall würde ein A erscheinen).

ACHTUNG: Hier muß etwas wichtiges über die sogenannten Bilddaten gesagt werden. Die Bildinformation steht als eingeschlossener gleichförmiger Block im Speicher. Es ist die Art von Arbeitszeile, auf die man die Daten unterbringt, die bestimmt, ob großer oder kleiner Text oder Graphikpixels erscheinen sollen.



Spätere Beispiele werden zeigen, wie mit Hilfe des LIST-Befehls das Programm Daten durch den Bereich mit der Bildinformation "fließen" läßt. Man beachte, wie verschieden die Interpretation der selben Elemente ausfällt, je nach programmiertem Gang.

Als nächstes kommen in der Displayliste einige Zweien. Die Zahlen sind die gewünschten OS-Gänge. Da wir uns in Graphik 0 befinden, benötigt die Diplayliste Zweien. Für jede dieser Zweien, einschließlich der in vorigen LMS-Zahl" verborgenden", läßt der Computer eine Arbeitszeile in Gang 0 (8 Pixels Höhe) anlaufen.

Nicht die Arbeitszeile der LMS- Instruktion vergessen !

Schließlich kommt die 65. Er ist in jeder DL enthalten. Die Tabelle gibt die Bedeutung dieser 65 an : "Gehe an der folgenden Adresse auf die DL zurück und warte auf den nächsten Bilddurchlauf."
Die 65 muß auf jeden Fall am Ende einer DL stehen, gleichgültig was danach kommt.

Nach diesen Grundinformationen können wir statt 0 andere Zahlen in Beispiel 1 eingeben (1-11). Man beachte jedoch, daß die Listung desto länger wird, je weniger Bildzeilen pro Arbeitszeile gebraucht werden. Also für einen Gang wie 7 mit nur zwei Bildzeilen Höhe, muß man viele Dreizehnen einfügen, um das Bild aufzufüllen.

Bei Eingabe von Gangzahl 8 erscheint in der Mitte der Dl eine 79. Auch am Anfang der DL steht bereits eine 79 als LMS-Zahl. Wie gesagt steht 79 für 64 + 15 (OS-Gang für Graphikgang 8). Wozu aber benötigt man eine zweite LMS- Zahl und einen zweiten Adressensatz für die Bilddaten ? Der Grund: Die Displayliste kann nur bis zu 4K (4096 Bytes) erreichen. Der zweite LMS setzt nach dem Ende des ersten Adressbereichs ein und weist auf Informationen im nächsten Speicherblock hin. Diese Grenzen sind durch die Hardware gegeben. Man muß sorgfältig rechnen, wenn man mehr als 4 K Bilddaten einsetzt.

Beispielprogramm 2:



Beispiel 2 dient dazu, das Auswechseln der DL zu üben. Die Zahlen in dem projezierten Daten-Statement sollen den Benutzer zur Ausgabe über Läufersteuerung anleiten. Zahlen können hier addiert, geändert oder gelöscht werden. Zur Kontrolle der eingegebenen Werte "RETURN" drücken.

Für den Fall, daß Verständigungsschwierigkeiten auftreten, sind bereits einige Zweien in Vieren geändert worden. Deshalb erscheint bei einfachen Druck auf "RETURN" ein Normalbild.

Jetzt wird klar, warum im Grundhandbuch einige der speziellen Gänge nicht erläutert werden: Gang 4 ist ein Vielfarben-Textgang für den Gebrauch bei Buchstaben-Graphik.

Er ist schwierig zu benutzen und wird daher auch nicht erklärt (siehe Spezialinformation).

Mit dem Beispielprogramm soll jetzt eine Weile geübt werden. Dabei bitte beachten, daß die Zahlen in der Displayliste die OS-Gangzahlen von 2-15 bedeuten.

Dies Programm ist bewußt einfachgehalten, damit der Gangwechsel gut geübt werden kann. Zum Überprüfen, welche Gänge gewählt wurden, verläßt man das Programm am besten durch BREAKE und tastet LIST ein. Die Listung läuft dann durch die einzelnen Gänge und zeigt, wo sich Text und Graphik befinden.

Beispielprogramm 3
==================

Dies ist eine kleine Subroutine, die jeder im eigenen Programm verwenden kann. Die Werte für die Displayliste kann man dem Beispiel 2 oder noch besser dem Beispiel 10 entnehmen. Wir haben der eigentlichen Routine einige Zeilen für den glatten Ablauf beigefügt. Beim Kopieren des Unterprogramms kann man diese weglassen und nur den Sprung in die Routine bei 700 sowie diese selbst stehen lassen (Zeile 10000 - 10040). - so arbeitet die Routine: Zeile 10010 setzt die beiden Teile der Adresse aus 560 und 561 zusammen. Der Abfabg der DL steht immer auf diesen Plätzen; da er jedoch variiert, sollte man diese Register stets prüfen, bevor man eine DL modifiziert.

Man muß den Wert prüfen, nachdem man einen Graphikgang angerufen hat. Der Anruf des Graphikgangs richtet dann automatisch eine DL und einen Bilddatenbereich im Speicher ein.

Der Anruf der Graphikgänge (0-11) geschieht wie folgt:
Wir benötigen dazu eine zweite Tabelle (Abb. 2). Sie enthält alle Standard- DL. Die erste zeigt normalen Textgang, Graphik 0. Die Listung fängt ca. 1 K unterhalb der Obergrenze des Speichers mit der eigentlichen Displayliste an. Dann kommen, in Richtung nach oben, die 960 Bytes Bilddaten. Beispiel 2 ruft Gang 0 an und ändert dann, nach Überprüfung von 560 und 561, die DL.

In diesem Beispiel genügen 960 Bytes für das Bild. Man soll immer nur soviel Speicherraum hierfür belegen, wie notwendig. Um Beurteilen zu können, mit welchem man starten soll, addiert man auf, wieviele Zahlen pro Gang man braucht, multipliziert jedes Ergebnis mit der Anzahl von Bytes pro Arbeitszeile und kommt auf eine Gesamtsumme.

Das nächste Fenster bedeutet eine leichte Komplikation für die DL. Nach der Folge von 100 Zwölfen, den LMS-Bytes und den verschiedenen OS-Gang-Codes findet man weitere LMS vor. Danach kommen wieder Arbeitszeilen (Text) und am Schluß nun die 65ger Adresse, die die DL beendet.

Die zusätzliche LMS-Instruktion besagt lediglich :"Rufe einige Daten aus einer anderen Speicherstelle als der augenblicklich adressierten und transportiere die in bestimmte Textgangzeilen (gewöhnlich 4 Zeilen von Gang 0)."

Beispielprogramme 4 und 5.

Beispiel 4 besteht im Einrichten einer Graphik-1-Displaylistung (Großbuchstaben), die jedoch in der Mitte eine zweite Speicherzeile läd (=Load Memory Scan = LMS). Die Daten hierfür lauten 70, 96 und 127, wobei die letzten beiden Zahlen jedoch in den Zeilen 10023-10024 versetzt werden, je nach Speicheranordnung.

Hier wird die Displayliste angewiesen , nach der Hälfte des Weges auf dem Monitor ihre Daten an der selben Stelle zu holen wie am

Anfang des Bildes. Man hätte hierfür auch irgendeinen anderen Speicherteil wählen können, denn das LMS- Byte legt einem wenig Beschränkung auf und kann nicht nur am Anfang der DL bzw. an den 4 Gang- Grenzen genutzt werden.

Beispiel 5 zeigt im Prinzip dasselbe, doch werden die eingetasteten Daten nur einmal projeziert. Beide Beispiele (4 und 5) könnten für Spiele benutzt werden, vor allem aber sind sie geeignet, die Programmierung von Großschrift für Sehbehinderte oder Kinder.

Zurück zu Beispiel 4 :
Das Programm meldet "fertig" (Ready) in Doppelbuchstaben. Sobald es gestoppt hat, kann man verschiedenes ausprobieren, z. B. eine Zeile eintasten wie folgt:

"Hallo, ich benutze Big-Buchstaben". Die Maschine wird dann so arbeiten, als wäre sie auf Graphikgang 0. Wieso, wird aus Zeile 720 des Beispiels 4 ersichtlich. Der POKE-Befehl auf die Position 87 täuscht den Computer. In der Subroutine eben zuvor wurde die DL in Graphikgang 1 angelegt, der Rechner sucht jedoch stets bei Adresse 87 nach , um festzustellen, in welchem Gang er arbeitet. Durch Einschieben von 0 (für Graphikgang 0) veranlaßt man ihn so zu listen, als wäre auf Gang 0.

Beispiel 5 arbeitet fast genauso, außer das mit Gang 0 begonnen wird antsatt mit dem POKE 87. Dadurch wird die Adresse 87 automatisch auf 0 gesetzt, wovon man sich durch PEEK 87 überzeugen kann.

Der zweite Unterschied liegt im Fehlen des zweiten LMS (Load Memory Scan,siehe oben) in der Mitte der Displaylistung. Beide Beispiele weisen großflächige Schrift auf, doch nimmt die Maschine immer noch an, auf Gang 0 zu stehen, und sie versucht, die normalen Buchstaben der Größe 40 mal 24 zu projizieren. Deshalb erscheint der Text nicht ganz vollständig. Man versuche es mit einem List-Befehl.

Anmerkung :
Um Beispiel 5 anzuwählen, muß man entweder RUN"DEX5" eintasten oder RUN"C:", entsprechendes Bild für Beispiel 6.

Beispiel 6:

Dies Beispiel ist dem Beispiel 2 ähnlich, nur das es die laufende DL im Gedächnis behält. Auf diese Weise kann man Schritt für Schritt die DL ändern, bis man zufrieden ist. Die Werte kann man dann kopieren und sie in Beispiel 3 für die jeweiligen eigenen Programme verwenden.

Nach Beendigung eines Bildaufbaus kann man durch BREAKE das Programm verlassen und LIST eingeben. Die Programmdaten fließen dann über die eben eingerichtete Displaylistung, und man kann die einzelnen Bereiche besser überblicken. Um neu zu starten braucht man nur RUN einzutasten.

Weitere Befehle : Durch Druck auf OPTION kommt man zu Beispiel 7, durch START läuft eine weitere Bildfolge ab, in der die laufenden DL-Werte benutzt werden; durch SELECT geht die Maschine auf die ursprünglichen Werte, was nützlich ist, falls man mit dem Programm Schwierigkeiten hat.

Solche Schwierigkeiten können z.B. auftreten, wenn zuviele Pixelzeilen (Bildzeilen) verbraucht werden und das Bild ins Rollen gerät, oder auch wenn zu wenige Zeilen programmiert werden, sodaß das Bild schrumpft.

Man muß dann das Verhältnis zwischen Bildmaterial und Zeilenverbrauch entsprechend ändern (wie oben beschrieben).

Beispielprogramm 7 :

Hierbei gilt es, sich selbst durchzufinden.
Hinweise : Sobald die "Hallos" aufhören, betrachte man die Gangangaben auf dem Monitor. Nahe dem unteren Ende entdeckt man Gang 8 (rote und blaue Pixels).

Da der Computer entsprechend den jeweils nächsten Daten "in der laufenden Reihe" arbeitet, hört die Wirkung der OPTION-, START- bzw. SELECT-Statements auf. Das Programm stößt dann auf eine 4 K - Grenze, sodaß eine neue LMS-Instruktion gesetzt wurde. Wir haben für diese jedoch die selbe Adresse hinsichtlich der Obergrenze der Daten benutzt wie zuvor, um die Sache interessanter zu machen.

Man beachte in diesem Zusammenhang die Displayliste für Graphikgang 8 im Beispiel 1, speziell die 79 (LMS) in der Mitte !

Beispielprogramm 8 :

Man muß sich hierbei daran erinnern, daß die LMS- Nummern (zwei Zahlen hinter dem LMS-Wert) dem Computer angeben, ehe er die Bilddaten nehmen soll. Beispiel 8 benutzt nun den Spielstab zur Änderung dieser Adresse, sosaß die meisten Speicherinhalte sowohl in Graphik- wie in Textform sichtbar werden, manchmal ist der Wechsel unmittelbar für den Benutzer sichtbar. Bewegt man den Stab vorwärts, so steuert er in Richtung des oberen Speicherteils, bewegt man ihn rückwärts, erreicht man fast das untere Ende des Speichers.

Würde man das äußerste Ende des Speichers ansteuern, so bräche das Programm zusammen, weil das OS zerstört würde. -

Beispielprogramm 9 :

Dies Programm zeigt das Aufsuchen und Ausdrucken von Material. Das Problem bei den üblichen Displaylisten besteht weniger im Wählen der Nummern, sondern, wie dies Beispiel zeigt, im Treffen der richtigen Plot- oder Print-Statements für die neu eir-erichtete Monitorszene.

Beim Start der Gebrauchs-DL mit einem POKE 87, Graphikgang Nummer X oder einem Graphik-Statement wie GR.0 nimmt das OS an, daß es sich im betreffenden Gang befindet und nicht in den Gängen, die man mit der Gebrauchs-DL aufs Bild gebracht hat.

Wenn die Bereiche, die geplottet bzw. angeschrieben werden, innerhalb der normalen Grenzen liegen, dann kann man ohne weiteres durch ausprobieren herausfinden, auf welche Arbeitszeile das PLOT bzw. das PRINT liegen soll. Eben dies geschieht im folgenden Beispiel. Mal trifft man, mal geht es daneben, doch man kommt immer ein Stück weiter. Man kann eine Position durch PLOT ansprechen und, wenn man nicht die richtige getroffen hat, das PLOT-Statement ändern. -

Im nächsten Beispiel wird nicht so verfahren, sondern es werden die ganzen Zahlenwerte für die Plazierungder Information auf dem Bild angegeben.

Beispielprogramm 10 :

In dieses Programm wurden alle zuvor einzeln geprobten Möglichkeiten eingebaut, und außerdem erscheinen die versprochenen Informationen auf dem Übersichtsbild.

Auch dies Programm verwendet Data-Statements zum experimentieren mit der Bildaufteilung in Kombination von Graphik - und Textfeldern.

Zum Ausgeben der Displayliste muß man auch hier die Läufersteuerung benutzen, durch RETURN sieht man, was man programmiert hat. Wenn PLOT-Gänge oben im Bild benutzt wurden, so sieht man das allerdings nicht ohne weiteres. Man muß also hier besonders aufpassen. Erscheint das gespaltene Normalbild, so hat man die Wahl zwischen

1. Druck auf START(wie zuvor), um die laufende DL zu ändern
2. Druck auf SELECT(wie zuvor),um zur ursprünglichen DL zurückzuspringen
3. Druck auf OPTION, um den Übersichtsplan der Normal - DL aufs Bild zu bekommen, die man soeben programmiert hat

Man wähle nun den neuen Übersichtsplan durch Druck auf RETURN und dann OPTION, wie oben erklärt. Die erste Spalte zeigt die Position jedes einzelnen Graphikgangs : 1,2,3... letzter Gang (max. 20). Die zweite Spalte zeigt, welche Zahl in Register 87 gebracht werden muß; es handelt sich einfach um die regulere Graphikgang-Nummer, die in der DL selbst angewand wurde. Spalte drei enthält die OS-Gangnummer, die in der DL selbst enthalten ist. Dies mag verwirrend erscheinen, da beide Zahlen dasselbe bedeuten, doch muß man bedenken, daß die normale Graphikgangnummer für den BASIC-Übersetzer durch diesen in die entsprechende OS - Nummer umgewandelt wird. Der Übersetzer muß aber eben umgangen werden, wenn man Sonderbilder programmieren will, die per BASIC machbar.

Die folgende Spalte zeigt die Anzahl von Zeilen des jeweils angerufenen Gangs, d.h. wie oft er in der DL eingesetzt ist. Das ist nützlich, weil man die Übersicht behält, wie weit ein bestimmter Gang wirksam ist, und es so vermeidet, durch einen zu weitreichenden PLOT aus dem betreffenden Gang herauszuspringen.

Dasselbe gilt für Textbereiche. Diese Spalte gibt eine schnelle Übersicht hierzu.

Es folgen die beiden Spalten mit den POKE - Zahlen für die Register 88 und 89. Diese beiden Werte dienen, in Kombination mit 87 dazu, der Anlage den Eindruck zu geben,als starte sie bei 0,0 oben links. Auf diese Weise arbeiten die PRINT- und PLOT-Statemente fast wie unter normalen Bedingungen. Man geht mit dem PRINTS und PLOTS für jeden einzelnen Bereich so vor, als begännen sie links auf 0,0.

Der projezierte Plan zählt außerdem auch die Bildzeilen zusammen, die man programmiert hat. Wenn man sieht, das die maximale Zeilenlänge noch nicht erreicht ist oder man etwas ändern will,kann man durch START die DL noch einmal ausdrucken lassen und die Zahl der Arbeitszeilen erhöhen bzw. kürzen.

Die Benutzung des hier besprochenden Programms geschieht am besten so, daß man die Teile ablaufen läßt, die man verstanden hat,und die anderen zunächst nicht anspricht. Die Punkte, die man unbedingt verstanden haben muß, sind:

1. Mit Beispiel 10 gelangt man aü zur Prüfung der DL, bü an die nach 87,88 und 89 zu transportierenden Zahlen.
2. Beispiel 11 ermöglicht die Umwandlung der POKE - Werte in die von Beispiel 10 gegebenen, nachdem die Zahlen von Beispiel 9 in die DL-Daten von Beispiel 10 übertragen worden sind.
3. Man muß versuchen, Graphik und Text wie sie in Beispiel 9 stehen, in eigene Version umzuwandeln.
4. Wenn man Bereiche wünscht als im Beispiel steht, sollte man einfach welche hinzufügen.
5. Man beachte, daß jeder Bildbereich so zu behandeln ist, als wäre er ein selbstständiger kleiner Monitor.- nicht über den jeweiligen Bereich hinaus gehen, wenn man PLOT - oder PRINT-Statements eingibt!

Beispielprogramm 11:

Es läuft in etwa wie Beispiel 9 ab, enthält aber zusätzliche POKE auf 88 und 89, um den Computer derart zu täuschen, daß er die Anfangsposition jedes Einzelgangs für 0,0 nimmt.

Der POKE an 87 sagt dann aus: "Arbeite so, als wärst Du in normalen Graphikgang (Nummer wie eingegeben)." Nun wird PLOT bzw. PRINT so eingesetzt, als sei jeder Bereich ein kleiner Bildschirm für sich. Nicht zu weit adressieren !

Der Code für das Beispiel wird beigefügt, weil es auf das Verständnis des Programms ankommt. Unsere übrigen Beispiele übernehmen gewissermaßen alle Rechenarbeit. Man beachte, daß der 5. und der 6. Zahlenwert in der Dataliste dem einzelnen Computer angepaßt werden muß (siehe oben).

Schlußbemerkung.

Beim Aufbau eines Gebrauchbildes kann es vorkommen, daß Text oder PLOT plötzllich auf die andere Bildhälfte rutschen. Das kommt daher, daß die DL zunächst z. B. in einem Gang mit 40 Bytes pro Zeile arbeitet, dann durch SHIFT etwa ein Stück weiter in einen Gang mit 20 Bytes pro Zeile gelangt. Das System versucht nach wie vor, die Daten in der alten Anordnung aufzureihen, was dann aber nicht mehr passt.

Man sollte bei Gangänderung also darauf achten, daß ein vielfaches der neuen Bytemenge jeweils wieder die alte Bytemenge ergibt, z.B. 4 mal 10 Byte = 40 Bytes oder 8 mal 10 Bytes = 2 mal 40 Bytes, die dann in 2 Schüben projeziert werden, falls ein 40-Bytegang programmiert worden war.

Es sei auch darauf hingewiesen, das Beispiel 10 gelegentlich durch ungewöhnliche Eingaben inkorrekt läuft. Dann muß man die Werte sorgfältig nachrechnen und neu eingeben.

```
1 REM BEISPIEL 1
2 REM
3 REM
4 REM
5 REM
1000 GRAPHICS 17:? #6;"  BEISPIEL7"
1010 FOR W=1 TO 230:POKE 710,W
1020 FOR ZR=1 TO 10:NEXT ZR
1030 NEXT W
1100 GRAPHICS 0
1110 DIM NUM(10):DIM A$(1):DIM M(200)
1112 FOR I=1 TO 9:READ X:NUM(I)=X:NEXT I
1114 DATA 32,34,24,34,54,54,94,94,176
1120 TRAP 40000:TRAP 1120:? "}":POSITION 1,5:? "WHICH GRAPHICS MODE WOULD YOU LIKE TO  SEE THE DISPLAY  LIST FOR";
1130 INPUT MODE
1140 IF MODE>8 OR MODE<0 THEN 1165
1150 GOTO 1220
1165 GRAPHICS 17:POKE 708,52
1170 POSITION 1,5:? #6;"MODES 0 TO 7 ONLY":SOUND 0,120,8,8:FOR W=1 TO 500:NEXT W:SOUND 0,0,0,0
1180 FOR W=1 TO 500:NEXT W:GOTO 1120
1220 USE=NUM(MODE+1)
1232 GRAPHICS MODE
1234 NON=PEEK(559):POKE 559,0
1240 J=1
1245 DL=PEEK(560)+256*PEEK(561)
1250 M(J)=PEEK(DL+J-1):J=J+1
1260 IF J<USE+1 THEN 1250
1261 GRAPHICS 0:K=1:POKE 559,NON
1263 ? M(K);"  ";:IF K>USE-1 THEN 1270
1265 K=K+1:GOTO 1263
1270 POSITION 2,20:? "WOULD YOU LIKE TO SEE ANOTHER MODE'S    DISPLAY LIST (Y OR N)";
1280 INPUT A$:IF A$="Y" THEN 1120
1290 IF A$="N" THEN STOP
1300 GOTO 1280
```



```
1 REM BEISPIEL 2
2 GRAPHICS 17:? #6;"  BEISPIEL 2"
3 FOR W=1 TO 230:POKE 710,W
4 FOR ZR=1 TO 10:NEXT ZR
5 NEXT W
7 GRAPHICS 0:DIM ZX$(135)
10 DL=PEEK(560)+256*PEEK(561)
15 A=PEEK(DL+4):B=PEEK(DL+5)
20 GRAPHICS 0:? "THIS EXAMPLE ALLOWS YOU TO CHANGE A   MODE 0 DISPLAY LIST!"
30 ? "PRESS START TO BEGIN"
55 DATA 112,112,112,66,64,156,2,2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,65,32,156,0,0,0,0
62 POKE 53279,8
65 Z=PEEK(53279)
70 IF Z=6 THEN 100
80 GOTO 65
100 TRAP 40000:? "}":POSITION 2,5
105 ? "55 DATA 112,112,112,66,";A;",";B;",2,2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,65,32,156,0,0,0,0"
110 POSITION 2,15:? "CHANGE THE VALUES YOU WANT BY USING   THE CURSOR CONTROLS-THEN PRESS RETURN TO SEE NEW ";
115 ? "SCREEN"
120 POSITION 0,1
125 INPUT ZX$
130 POSITION 0,5
140 POKE 842,13
142 POSITION 2,12:? "CONT"
144 POSITION 0,3:STOP
150 POKE 842,12
200 RESTORE 55
210 I=0
215 TRAP 100
220 READ G:POKE DL+I,G:I=I+1:IF G=0 THEN 235
230 GOTO 220
235 FOR W=1 TO 900:NEXT W:LIST
240 ? "PRESS OPTION TO GO ON":? "PRESS START TO DO ANOTHER SCREEN"
245 TRAP 100
250 POKE 53279,8
260 Z=PEEK(53279)
270 IF Z=6 THEN 57
280 IF Z=3 THEN STOP
300 GOTO 260
1000 ? PEEK(DL+I):? " ";DL+I:I=I+1:GOTO 1000
```

```
1 REM BEISPIEL 3
2 REM
3 REM
10 GRAPHICS 17:? #6;"  BEISPIEL?"
20 FOR W=1 TO 230:POKE 710,W
30 FOR ZR=1 TO 10:NEXT ZR
40 NEXT W
500 REM THIS IS A SUBROUTINE FOR YOU TO USE IN YOUR OWN PROGRAMS. SEE MANUAL FOR DIRECTIONS
700 GOSUB 10000
710 REM PUT THE REST OF YOUR PROGRAM STARTING ON THIS LINE
715 FOR W=1 TO 2000:NEXT W
720 GRAPHICS 0:? "THIS EXAMPLE IS FOR YOU TO USE IN YOUR PROGRAMS":? "PRESS BREAK AND SAVE IT IF YOU WANT OR PRESS START";
730 ? " TO GO ON":POKE 53279,8
740 Z=PEEK(53279)
750 IF Z=6 THEN STOP
760 GOTO 740
10000 TRAP 10040
10010 DL=PEEK(560)+256*PEEK(561)
10011 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
10012 I=1
10013 READ A:IF A=0 THEN 10040
10014 POKE (DL+I-1),A
10015 IF I=5 THEN POKE DL+4,Z1
10016 IF I=6 THEN POKE DL+5,Z2
10020 I=I+1:GOTO 10013
10030 DATA 112,112,112,66,64,156,2,2,2,5,5,5,5,2,2,4,4,4,4,4,2,65,32,124,0,0,0
10040 RETURN
```

```
1 REM BEISPIEL 4
10 GRAPHICS 17:? #6;" BEISPIEL4"
20 FOR W=1 TO 230:POKE 710,W
30 FOR ZR=1 TO 10:NEXT ZR
40 NEXT W
490 REM ******************************
500 REM HERE IS A PRACTICAL EXAMPLE OFWHAT YOU CAN DO WITH A DISPLAY LIST   THAT IS MODIFIED
510 REM JUST SAY GOTO 700    WHEN YOU  WANT TO SEE YOUR PROGRAM IN BIG       LETTERS AND IN DUPLICATE!!
520 REM ******************************
700 GOSUB 10000
710 REM YOUR BASIC PROGRAM GOES HERE
720 POKE 87,0:END
10000 REM TRAP 10040:GRAPHICS 1
10005 GRAPHICS 1
10010 DL=PEEK(560)+256*PEEK(561)
10015 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
10016 I=1
10017 READ A:IF A=0 THEN 10040
10020 POKE (DL+I-1),A
10021 IF I=5 THEN POKE DL+4,Z1-32
10022 IF I=6 THEN POKE DL+5,Z2+2
10023 IF I=16 THEN POKE DL+15,Z1-32
10024 IF I=17 THEN POKE DL+16,Z2+2
10025 I=I+1:GOTO 10017
10030 DATA 112,112,112,70,96,127,6,6,6,6,6,6,6,6,70,96,127,6,6,6,6,6,6,6,6,6,65,94,125,0,0
10040 RETURN
```



```
1 REM BEISPIEL 5
10 GRAPHICS 17:? #6;"  BEISPIEL 5"
20 FOR W=1 TO 230:POKE 710,W
30 FOR ZR=1 TO 10:NEXT ZR
40 NEXT W
500 REM THIS IS A SUBROUTINE TO ALLOW YOU TO PROGRAM WITH BIG LETTERS.SEE   MANUAL FOR DIRECTIONS.
700 GRAPHICS 0:GOSUB 10000
710 REM PUT THE REST OF YOUR PROGRAM  STARTING ON THIS LINE
720 LIST :END
10000 TRAP 10040
10010 DL=PEEK(560)+256*PEEK(561)
10011 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
10012 I=1
10013 READ A:IF A=0 THEN 10040
10014 POKE (DL+I-1),A
10015 IF I=5 THEN POKE DL+4,Z1+70
10016 IF I=6 THEN POKE DL+5,Z2+2
10020 I=I+1:GOTO 10013
10030 DATA 112,112,112,70,74,158,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,65,32,124,0,0,0
10040 RETURN
```

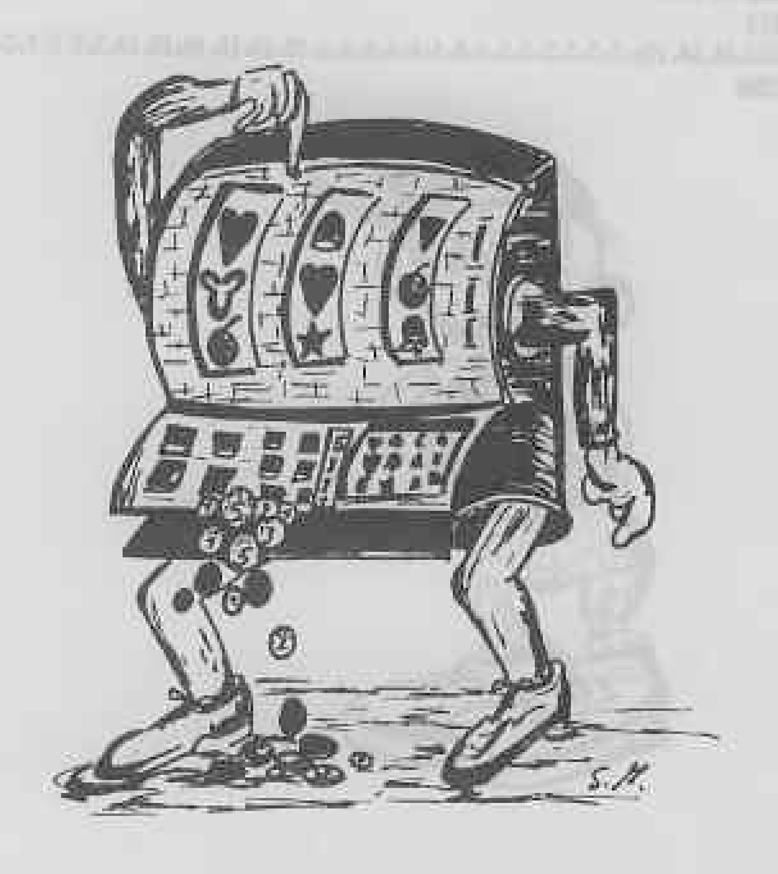
```
1 GRAPHICS 17:? #6;"  BEISPIEL6"
3 FOR W=1 TO 230:POKE 710,W
4 FOR ZR=1 TO 10:NEXT ZR
5 NEXT W
7 DIM ZX$(135)
20 GRAPHICS 0:? "THIS EXAMPLE ALLOWS YOU TO CHANGE A   MODE 0 DISPLAY LIST!"
30 ? "PRESS START TO BEGIN":? "WRITE DOWN THE NUMBERS IN THE DL WHEN YOU LIKE IT!"
55 DATA 112,112,112,66,96,144,2,2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,2,65,32,156,0
56 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
57 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
58 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
62 POKE 53279,8
65 Z=PEEK(53279)
70 IF Z=6 THEN 100
80 GOTO 65
100 GRAPHICS 0:POSITION 2,3
105 LIST 55,58
110 POSITION 2,18:? "CHANGE THE VALUES YOU WANT BY USING   THE CURSOR CONTROLS "
115 ? "SCREEN."
120 POSITION 10,0
125 INPUT ZX$
130 POSITION 0,2
140 POKE 842,13
142 POSITION 2,17:? "CONT"
144 POSITION 0,2:STOP
150 POKE 842,12
200 RESTORE 55
210 I=0
215 TRAP 400
217 I=1:GRAPHICS 7+16
218 DL=PEEK(560)+256*PEEK(561)
219 A=PEEK(DL+4):B=PEEK(DL+5)
220 READ G:IF G=0 THEN 233
221 POKE DL+I-1,G
222 IF I=5 THEN POKE DL+4,A
223 IF I=6 THEN POKE DL+5,B
230 I=I+1:GOTO 220
233 FOR W=1 TO 900:NEXT W:POKE 87,0
240 POSITION 2,1:? "PRESS OPTION TO GO ON":? "PRESS START TO DO ANOTHER SCREEN"
243 ? "PRESS SELECT FOR THE ORIGINAL DISPLAY LIST"
245 TRAP 5
250 POKE 53279,8
260 Z=PEEK(53279)
265 IF Z=5 THEN 400
270 IF Z=6 THEN 57
280 IF Z=3 THEN STOP
300 GOTO 260
400 TRAP 400:GRAPHICS 0:POSITION 2,4
405 ? "55 DATA 112,112,112,66,";A;",";B;",2,2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,2,65,32,156,0"
406 ? "56 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
407 ? "57 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
408 ? "58 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
410 GOTO 110
```

```
1 REM BEISPIEL 7
2 GRAPHICS 17:? #6;"  BEISPIEL7"
4 FOR W=1 TO 230:POKE 710,W
5 FOR ZR=1 TO 10:NEXT ZR
6 NEXT W
7 DIM ZX$(135)
10 TRAP 40000:TRAP 10
20 GRAPHICS 0:? "THIS EXAMPLE ALLOWS YOU TO CHANGE A   MODE 0 DISPLAY LIST!"
30 ? "PRESS START TO BEGIN":? "WRITE DOWN THE NUMBERS IN THE DL WHEN YOU LIKE IT!"
40 POKE 710,0
55 DATA 112,112,112,66,96,144,2,2,2,2,6,6,7,7,14,14,14,14,14,14,14,2,2,15,15
56 DATA 15,15,15,15,15,79,96,144,15,6,7,2,2,65,32,156,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
57 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
58 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
62 POKE 53279,8
65 Z=PEEK(53279)
70 IF Z=6 THEN 100
80 GOTO 65
100 ? "}":POSITION 2,3
105 LIST 55,58
110 POSITION 2,18:? "CHANGE THE VALUES YOU WANT BY USING   THE CURSOR CONTROLS "
115 ? "SCREEN."
120 POSITION 10,0
125 INPUT ZX$
130 POSITION 0,2
140 POKE 842,13
142 POSITION 2,17:? "CONT"
144 POSITION 0,2:STOP
150 POKE 842,12
200 RESTORE 55
210 I=0
215 TRAP 400
217 I=1:GRAPHICS 7+16:POKE 710,0
218 DL=PEEK(560)+256*PEEK(561)
219 A=PEEK(DL+4):B=PEEK(DL+5)
220 READ G:IF G=0 THEN 233
221 POKE DL+I-1,G
222 IF I=5 THEN POKE DL+4,A
223 IF I=6 THEN POKE DL+5,B
230 I=I+1:GOTO 220
233 FOR W=1 TO 900:NEXT W:POKE 87,0:LIST :GOSUB 5000
240 ? "PRESS OPTION TO GO ON":? "PRESS START TO DO ANOTHER SCREEN"
243 ? "PRESS SELECT FOR THE ORIGINAL DISPLAY LIST"
245 TRAP 10
250 POKE 53279,8:J=0
260 Z=PEEK(53279)
265 IF Z=5 THEN 400
270 IF Z=6 THEN 57
280 IF Z=3 THEN STOP
300 GOTO 260
400 TRAP 40000:GRAPHICS 0:POSITION 2,4
405 ? "55 DATA 112,112,112,66,";A;",";B;",2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,65,32,156,0"
406 ? "56 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
407 ? "57 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
408 ? "58 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
410 GOTO 110
5000 J=J+1:? "HELLO":IF J<50 THEN 5000
5010 RETURN
```

```
1 REM BEISPIEL 8
10 GRAPHICS 17:? #6;"  BEISPIEL8"
20 FOR W=1 TO 230:POKE 710,W
30 FOR ZR=1 TO 10:NEXT ZR
40 NEXT W
500 REM THIS IS A SUBROUTINE FOR YOU TO USE TO LOOK AT MEMORY BOTH FROM A GRAPHICS AND TEXTUAL VIEWPOINT!
520 REM USE YOUR JOYSTICK AND MOVE UP OR DOWN IN MEMORY !!! PRESS START TO GO ON.
700 GRAPHICS 0:GOSUB 10000
705 I=Z1:LIST :POKE 53279,8
707 FOR F=1 TO 50:? "USE JOYSTICK":NEXT F
710 ST=STICK(0):Z=PEEK(53279):IF Z=6 THEN RUN "D:EX9"
715 IF ST=15 THEN 710
720 IF ST=14 THEN Z2=Z2+1
730 IF ST=13 THEN Z2=Z2-1
740 IF Z2<10 THEN Z2=10
745 IF Z2>PEEK(106) THEN Z2=PEEK(106)
750 POKE DL+5,Z2
760 GOTO 710
10000 TRAP 10040
10010 DL=PEEK(560)+256*PEEK(561)
10011 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
10012 I=1
10013 READ A:IF A=0 THEN 10040
10014 POKE (DL+I-1),A
10015 IF I=5 THEN POKE DL+4,Z1
10016 IF I=6 THEN POKE DL+5,Z2
10020 I=I+1:GOTO 10013
10030 DATA 112,112,112,66,64,156,2,2,2,2,2,2,6,6,6,6,6,6,6,6,10,10,10,10,10,10,2,2,2,2,2,65,32,124,0,0,0
10040 TRAP 40000:RETURN
```



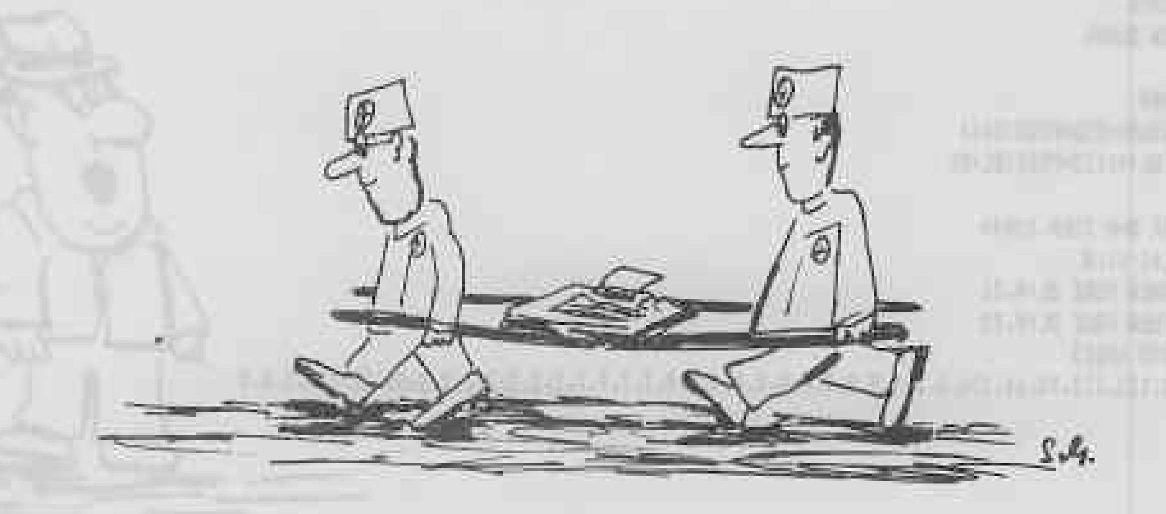
```
1 REM BEISPIEL 9
10 GRAPHICS 17:? #6;"  BEISPIEL 9"
20 FOR W=1 TO 230:POKE 710,W
30 FOR ZR=1 TO 10:NEXT ZR
40 NEXT W
500 REM THIS IS A SUBROUTINE FOR YOU TO USE IN YOUR OWN PROGRAMS. SEE MANUAL FOR DIRECTIONS.
505 GRAPHICS 0
510 GOSUB 10000
515 POKE 87,1:POSITION 0,0:? #6;"THIS IS BIG LETTERS"
520 ? #6;"ON TWO LINES":? #6;"OR THREE"
550 POKE 87,4:COLOR 1:PLOT 1,5:DRAWTO 1,15:DRAWTO 60,15:PLOT 15,15:DRAWTO 15,10:PLOT 3,15:DRAWTO 3,8
555 PLOT 5,15:DRAWTO 5,6:PLOT 7,15:DRAWTO 7,12:PLOT 9,15:DRAWTO 9,8:PLOT 11,15:DRAWTO 11,9:PLOT 13,15:DRAWTO 13,11
560 POKE 87,2:POSITION 0,9:? #6;"MOREBIG LETTERS ,in fact another three   rows of em"
570 POKE 87,0:POSITION 1,6:? "AND OF COURSE SOME TINY TEXT....PRESS   START TO GO ON"
730 POKE 53279,8
740 Z=PEEK(53279)
750 IF Z=6 THEN STOP
760 GOTO 740
10000 TRAP 10040
10010 DL=PEEK(560)+256*PEEK(561)
10011 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
10012 I=1
10013 READ A:IF A=0 THEN 10040
10014 POKE (DL+I-1),A
10015 IF I=5 THEN POKE DL+4,Z1
10016 IF I=6 THEN POKE DL+5,Z2
10020 I=I+1:GOTO 10013
10030 DATA 112,112,112,70,64,156,6,6,9,9,9,9,9,9,9,9,9,9,7,7,7,7,2,2,2,2,2,265,32,124,0,0,0
10040 RETURN
```



```
1 GRAPHICS 17:? #6;" BEISPIEL 10"
2 FOR W=1 TO 230:POKE 710,W
3 FOR ZR=1 TO 10:NEXT ZR
4 NEXT W
5 DIM P(220):DIM NUM(20):DIM MODE(20):DIM RAN(15):DIM START(20):DIM PB8(20):DIM PB9(20):DIM LIN(15)
7 DIM BAN(15)
10 TRAP 490
20 DIM ZX$(135)
30 GRAPHICS 0:? "THIS EXAMPLE ALLOWS YOU TO CHANGE A   MODE 0 DISPLAY LIST!"
40 ? "PRESS START TO BEGIN.":? "WRITE DOWN THE NUMBERS IN THE DL WHEN YOU LIKE THEM!"
50 DATA 112,112,112,66,64,156,2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,65,32,156,0
60 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
70 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
80 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0
90 POKE 53279,8
100 Z=PEEK(53279)
110 IF Z=6 THEN 140
120 GOTO 100
130 TRAP 490
140 GRAPHICS 0:POSITION 2,3
150 LIST 50,80
160 POSITION 2,18:? "CHANGE THE VALUES YOU WANT BY USING   THE CURSOR CONTROLS."
170 ? "PRESS RETURN TO SEE THE NEW SCREEN."
180 POSITION 10,0
190 INPUT ZX$
200 POSITION 0,2
210 POKE 842,13
220 POSITION 2,17:? "CONT"
230 POSITION 0,2:STOP
240 POKE 842,12
250 RESTORE 50:MAX=0
260 TRAP 490:READ S:IF S>15 THEN 260
265 IF S>MAX THEN MAX=S
270 IF S=0 THEN 281
280 GOTO 260
281 ON MAX GOTO 285,285,285,285,285,285,285,285,285,285,286,287,287,288,288
285 USE=0:GOTO 290
286 USE=22:GOTO 290
287 USE=23:GOTO 290
288 USE=24
290 RESTORE 50
300 TRAP 490
310 I=1:GRAPHICS USE
320 DL=PEEK(560)+256*PEEK(561)
330 A=PEEK(DL+4):B=PEEK(DL+5)
340 READ G:IF G=0 THEN 390
```

```
350 POKE DL+I-1,G
360 IF I=5 THEN POKE DL+4,A
370 IF I=6 THEN POKE DL+5,B
380 I=I+1:GOTO 340
390 FOR Q=1 TO 100:NEXT Q:POKE 87,0
400 POSITION 2,1:? "PRESS OPTION FOR CHART OF PLOT VALUES WHEN DONE."
410 ? "PRESS SELECT FOR THE ORIGINAL DISPLAY LIST #'S."
420 ? "PRESS START TO CHANGE CURRENT ▸7      NUMBERS."
425 ? "PRESS START & OPTION TO RUN LAST      EXAMPLE."
430 POKE 53279,8
440 Z=PEEK(53279)
450 IF Z=5 THEN 490
460 IF Z=6 THEN 70
465 IF Z=2 THEN STOP
470 IF Z=3 THEN 550
480 GOTO 440
490 TRAP 40000:GRAPHICS 0:POSITION 2,4
500 ? "50 DATA 112,112,112,66,";A;",";B;",2,2,2,2,2,2,2,2,2,4,4,4,4,4,2,2,2,2,2,2,2,2,2,65,32,156,0"
510 ? "60 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
520 ? "70 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
530 ? "80 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,65,32,156,0"
540 GOTO 160
550 RESTORE 50:POKE 559,0
560 J=1:K=1:L=2:P(1)=PEEK(DL+3)-64
570 READ PR:P(L)=PR:IF P(L)>15 THEN 570
580 IF P(L)=0 THEN 620
590 IF P(L)<>P(L-1) THEN MODE(K)=P(L-1):NUM(K)=J:J=1:L=L+1:K=K+1:GOTO 570
600 J=J+1:L=L+1:GOTO 570
620 MODE(K)=P(L-1):NUM(K)=J
630 RAM(2)=40:RAM(3)=40:RAM(4)=40:RAM(5)=40:RAM(13)=40:RAM(14)=40:RAM(15)=40
633 LIN(2)=8:LIN(3)=10:LIN(4)=8:LIN(5)=16:LIN(13)=2:LIN(14)=1:LIN(15)=1
634 LIN(6)=8:LIN(7)=16:LIN(8)=8:LIN(9)=4:LIN(10)=4:LIN(11)=2:LIN(12)=1
635 RAM(6)=20:RAM(7)=20:RAM(8)=10:RAM(9)=10:RAM(10)=20:RAM(11)=20:RAM(12)=20
640 FOR H=1 TO K
644 BAM(2)=0:BAM(3)=40:BAM(4)=40:BAM(5)=40:BAM(13)=7:BAM(14)=40:BAM(15)=8
645 BAM(6)=1:BAM(7)=2:BAM(8)=3:BAM(9)=4:BAM(10)=5:BAM(11)=6:BAM(12)=40
660 IF MODE(H)<MODE(H-1) THEN 680
670 SAV=MODE(H)
680 NEXT H
690 IF SAV>6 THEN 720
700 IF SAV>0 THEN SAV=6:GOTO 720
710 IF SAV<>0 THEN ? "ERROR AT 710..SAV SHOULD BE 0":STOP
720 MAXRAM=RAM(SAV)
725 FOR M=1 TO K:M2=0:M3=0
735 FOR S=1 TO M-1
740 IF RAM(MODE(S))=20 THEN 760
750 IF RAM(MODE(S))=10 THEN 770
755 GOTO 780
760 M2=NUM(S)+M2:GOTO 780
770 M3=NUM(S)+M3
780 NEXT S
783 R=0
```

```
785 FOR W=1 TO M-1:R=R+NUM(W):NEXT W
790 CHRPOS=A+B*256+(R*MAXRAM-M2*(MAXRAM-20)-M3*(MAXRAM-10))
795 IF M=1 THEN CHRPOS=A+B*256
800 START(M)=CHRPOS
810 NEXT M
820 FOR Z=1 TO K:P89(Z)=INT(START(Z)/256):P88(Z)=START(Z)-P89(Z)*256:NEXT Z
825 SUM=0
830 FOR Q=1 TO K:SUM=NUM(Q)*LIN(MODE(Q))+SUM:NEXT Q
840 GRAPHICS 0:POSITION 0,0:? "**********************************************"
850 POSITION 0,1:? "*POS*POKE*O.S. *# LINES* POKE * POKE *"
860 POSITION 0,2:? "*ON *INTO*MODE *OF THIS* INTO * INTO *"
870 POSITION 0,3:? "*SCR* 87 * #   * MODE  *  88  *  89  *"
880 POSITION 0,4:? "**************************************"
890 FOR W=1 TO K
895 SD=BAM(MODE(W))
898 POSITION 1,4+W:? W;
901 IF SD=40 THEN 903
902 POSITION 6,4+W:? SD;:GOTO 904
903 POSITION 6,4+W:? "NONE";
904 POSITION 12,4+W:? MODE(W):POSITION 18,4+W:? NUM(W):POSITION 25,4+W:? P88(W);
905 POSITION 33,4+W:? P89(W)
910 NEXT W:? "THE NUMBER OF PIXEL LINES USED=";SUM;".":? "SUGGEST YOU CHANGE DL UNTIL 192 PIXEL LINES RESULTS!";
915 ? " COPY NUMBERS OR PRESS  START TO REDO CURRENT DL.":? "GRAPHICS MODE TO USE IN EX.11 IS ";USE
920 POKE 559,34
930 POKE 53279,8
940 Z=PEEK(53279)
950 IF Z=6 THEN 140
960 GOTO 940
999 END
```



```
1 REM BEISPIEL 11
10 GRAPHICS 17:? #6;"  BEISPIEL 11"
20 FOR W=1 TO 230:POKE 710,W
30 FOR ZR=1 TO 10:NEXT ZR
40 NEXT W
60 FOR W=1 TO 230:POKE 708,W:POKE 709,W:POKE 710,W:POKE 711,W
65 FOR ZR=1 TO 5:NEXT ZR
70 NEXT W
90 ? #6;"WORK, YOU WILL HAVE "
100 ? #6;"TO CHANGE THE VALUES"
110 ? #6;"EACH TIME POKE 88 OR"
120 ? #6;"89 IS USED. THIS IS "
130 ? #6;"DONE BY INPUTING THE"
140 ? #6;"VALUES IN THE DL OF "
150 ? #6;"THIS EX. INTO BS10-."
160 ? #6;"the chart will give "
170 ? #6;"you the values for  "
180 ? #6;"each mode area.break"
190 ? #6;"now and look at DL  "
200 POKE 764,255
210 IF PEEK(764)<>12 THEN 210
400 REM *******************************
500 REM THIS IS A SUBROUTINE FOR YOU TO USE AS A PRACTICE EXAMPLE IN PLOTTING AND WRITING TO THE AREAS OF THE
503 REM SCREEN THAT YOU SET UP IN THE PREVIOUS EXAMPLE. JUST PLUG IN THE    POKE VALUES FOR LOCATIONS 87,88,&89
504 REM AND CHANGE THE NUMBERS IN THE DATA STATEMENT TO THOSE YOU WROTE DOWN FROM THE LAST EXAMPLE . FINALLY,YOU
505 REM SHOULD CHANGE THE PLOT AND PRINT STATEMENTS TO THE ONES YOU WANT. HAVE FUN...SEE YOU IN THE NEXT TUTORIAL.
506 REM *******************************
507 GRAPHICS 0:REM PUT 0-8 HERE, REMEMBER TO USE THE GRAPHICS MODE THAT THE CHART GAVE YOU.
510 GOSUB 10000
515 POKE 87,1:POKE 88,64:POKE 89,156:POSITION 0,0:? #6;"THIS IS BIG LETTERS"
520 ? #6;"ON TWO LINES":? #6;"OR THREE"
540 POKE 88,124:POKE 89,156
550 POKE 87,4:COLOR 1:PLOT 1,0:DRAWTO 1,10:DRAWTO 60,10:PLOT 15,10:DRAWTO 15,5:PLOT 3,10:DRAWTO 3,3
555 PLOT 5,10:DRAWTO 5,1:PLOT 7,10:DRAWTO 7,7:PLOT 9,10:DRAWTO 9,3:PLOT 11,10:DRAWTO 11,4:PLOT 13,10:DRAWTO 13,6
557 POKE 88,234:POKE 89,156
560 POKE 87,2:POSITION 0,0:? #6;"MOREBIG  LETTERS,in fact another thr"
565 POKE 88,58:POKE 89,157
570 POKE 87,0:POSITION 0,0:? "AND OF COURSE SOME TINY TEXT....PRESS   RETURN TO GO ON"
730 POKE 53279,8
740 Z=PEEK(53279)
750 IF Z=6 THEN 20000
760 GOTO 740
10000 TRAP 10040
10010 DL=PEEK(560)+256*PEEK(561)
10011 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
10012 I=1
10013 READ A:IF A=0 THEN 10040
10014 POKE (DL+I-1),A
10015 IF I=5 THEN POKE DL+4,Z1
10016 IF I=6 THEN POKE DL+5,Z2
10020 I=I+1:GOTO 10013
10030 DATA 112,112,112,70,64,156,6,6,9,9,9,9,9,9,9,9,9,9,9,7,7,7,7,7,2,2,2,2,2,2,2,65,32,124,0,0,0
10040 RETURN
```

II. Beschreibung der Hardware.

A. Die Chips ANTIC und CTIA.

1. Zum TV-Monitor:

Die Chips ANTIC und CTIA erzeugen die Bildzeichnung mit einer Frequenz von 5o Durchläufen pro Sekunde (im PAL-System).

Jeder Durchlauf des Bildstrahls besteht aus 31o Zeilen, und jede Einzelzeile ist in 228 Abschnitte, sog. Color-Clocks eingeteilt (s. Abb.VI-3). Der 65o2-Mikroprozessor läuft mit 1,79 MHZ. Diese Frequenz wurde gewählt, damit ein Maschinenzyklus genau 2 Color-Clocks entspricht. Ein solches Clock hat etwa die Breite von zwei Bildzeilen-Höhen.

In jedem Graphikgang wird der Bildschirm in kleine Rechtecke aufgeteilt, die man "Pixels" (Bildelemente) nennt. Der Graphikgang mit der höchsten Auflösung hat eine Pixelgrösse von 1/2 Colorclock x 1 Zeilenhöhe.

Die laufende Zeile kann durch Ablesung des Vertikal-Zählregisters (VCOUNT) festgestellt werden. Dies Register gibt die Anzahl der durchlaufenen Zeilen geteilt durch 2 an. Ein Durchlauf (PAL) hat 31o Zeilen, so dass das VCOUNT von o - 155 zählt.

Der Punkt Null tritt nahe am sogen. Vertikal-Leertakt ein. Dieser Takt entspricht der Zeit, während der der Bildstrahl vom Bildende nach links oben zum Beginn des nächsten Bilddurchlaufs zurückwandert.

Beim System ATARI 8oo gibt es keinen Bildeinschub (Interlacing), so dass der Bilddurchlauf stets genau gleich bleibt, solange das jeweilige Programm ihn nicht ändert. Der Punkt der Vertikal-Synchronisation (VSYNC) liegt innerhalb der Höhe 4-6 während des Vertikal-Leertaktes (VCOUNT = hexadezimal 70 - 7F). Dieser Schaltpunkt meldet dem TV-Gerät, wo der jeweilige Bilddurchlauf beginnt. Hinter dem VSYNC liegen noch 16 weitere Zeilen des Vertikal-Leertaktes, so dass im Ganzen 22 Taktzeilen zur Verfügung stehen. Die Sprung- bzw. Warte-Instruktion (s.u.) veranlasst, dass die Graphikabläufe der Display-Listung erst am Ende des Vertikal-Leertaktes beginnen.

2. Zum Operationssystem (OS):

Die ATARI 4oo/8oo wird mit einem 1o-K-Operationssystem (OS) im ROM geliefert. Das OS beansprucht einige der Hardwareregister, so dass es in dieser Beschreibung ab und zu erwähnt werden muss. Für detaillierte Information gibt es ein OS-Handbuch.

Das OS läuft bei den meisten Hardware-Gängen mit (BASIC, GRAPHICS, PLOT und DRAWTO sind die entsprechenden Befehle). Das OS liefert automatisch 24 Hintergrundzeilen nach dem Beenden des Vertikal-Leertaktes. Dadurch wird sichergestellt, dass die gegebenenfalls vorhandenen Randüberzeichnungen des TV-Gerätes (overscan) ausgeglichen werden. Die meisten Fernsehgeräte sind so gebaut, dass sie die Bildkanten abschneiden. Das wirkt sich verbessernd aus beim normalen Fernsehbetrieb, jedoch muss beim Computereinsatz die Information ganz vollständig aufs Bild gelangen. Gewöhnlich beträgt die Randüberschreitung 4 - 8 Colorclocks an beiden seitlichen Rändern. Ein TV-Gerät, das übermässigen Overscan hat, muss ggfls. justiert werden, damit es gute Ergebnisse liefert.

Das OS benutzt 192 Zeilen für seinen Bildaufbau und 24 Zeilen für den Overscan. Es nimmt die Standardbreite von 17o Colorclocks ein. Die Hardware erlaubt eigentlich Darstellungen beliebiger Länge, man sollte sich aber an diese Standards halten. Ausnahmen mögen bei nur dekorativer, nicht programmrelevanter Information gemacht werden.

3.Zur Schattenfunktion des OS:
Da viele der Hardwareregister nur schreibend, also nicht ablesbar sind, besitzt das OS eine Anzahl von sogen. Schattenregistern im RAM. Bei jedem Bilddurchlauf, und zwar während des Vertikal-Leertaktes, übernimmt das OS die Werte einiger Schattenregister und schreibt sie in das entsprechende Hardwareregister. Das OS schaltet einen Farbwechsel auf allen Farbregistern ein, wenn die A - TRACT-Information (im OS-Register) negativ ist. Dadurch wird eine Beschädigung des Leuchtbelags auf dem Schirm durch zu lange und zu helle stehende Bilder vermieden.

Das OS liest auch die Werte der Spielhebel bzw. der anderen Aussenkontrollgeräte während des Vertikal-Leertaktes ein und stellt die jeweiligen Werte im Schattenregister ab, so dass Benutzerprogramme nicht eigens diese Daten ermitteln müssen.

Es sind weiter einige unterbrechungsabhängige Register vorhanden, welche das OS während Unterbrechungsvorgängen ändert oder abliest. Programme fragen gewöhnlich die Schattenregister und nicht die Hardwareregister ab.

Die Schattenfunktion des OS kann allerdings abgeschaltet werden, indem man die Vertikal-Leer- und Unterbrechungs-Vektoren ändert (s. hierzu OS-Handbuch).

4. Zum WSYNC:
Zusätzlich zum Vertikal-Leertakt, der dem Mikroprozessor eine Vertikal-Synchronisation mit dem Bildablauf ermöglicht, liefert dies System auch einen Befehl "Warte auf Horizontal-Synchronisation" (WSYNC), durch welchen sich der Rechner automatisch an den horizontalen Zeilenablauf des Monitors anpassen kann.

Diese Synchronisation tritt in Kraft, sobald der Rechner bei entsprechender Notwendigkeit ein Input/Output-Register mit der Bezeichnung WSYNC anspricht. Beim Anschreiben dieser Adresse wird eine Sperre gerastet, die ein Signal mit der Bezeichnung "READY" im Mikroprozessor auf Nullstellung bringt. Sobald die READY-Funktion auf Null steht, stoppt der Rechner und wartet. Die Sperre wird automatisch ausgerastet (auf READY zurückgestellt), sobald das nächste Horizontal-Leer-Intervall beginnt, so dass dann der Rechner seine Programmarbeit wieder aufnimmt.

5. Zum Objekt-DMA (Direkte Speicheradressierung):
Die Hauptfunktion des ANTIC-Chips liegt im Holen von Daten aus dem Hauptspeicher (unabhängig vom Mikroprozessor), die auf den Monitor gebracht werden sollen.
Dies läuft über eine Technik ab, die sich Direkte Speicheradressierung (Direct Memory Access) nennt, DMA.
DMA fordert die Benutzung der Speicheradresse bzw. der Übertragungskanäle durch Aussenden eines HALT-Signals an den Mikroprozessor an. Dadurch wird dieser für den gesamten folgenden Maschinenzyklus in einen Zustand gebracht, der sich TRISTATE (offener Stromkreis) nennt.

Das ANTIC-Chip übernimmt dann den Adresskanal und liest alle gewünschten Daten aus dem Speicher. Ein anderer Name für diese Art DMA ist Zyklus-"Stehlen". Einmal angeschaltet, wird diese DMA vollständig und automatisch vom ANTIC-Chip her kontrolliert, ohne dass der Mikroprozessor eingreift.

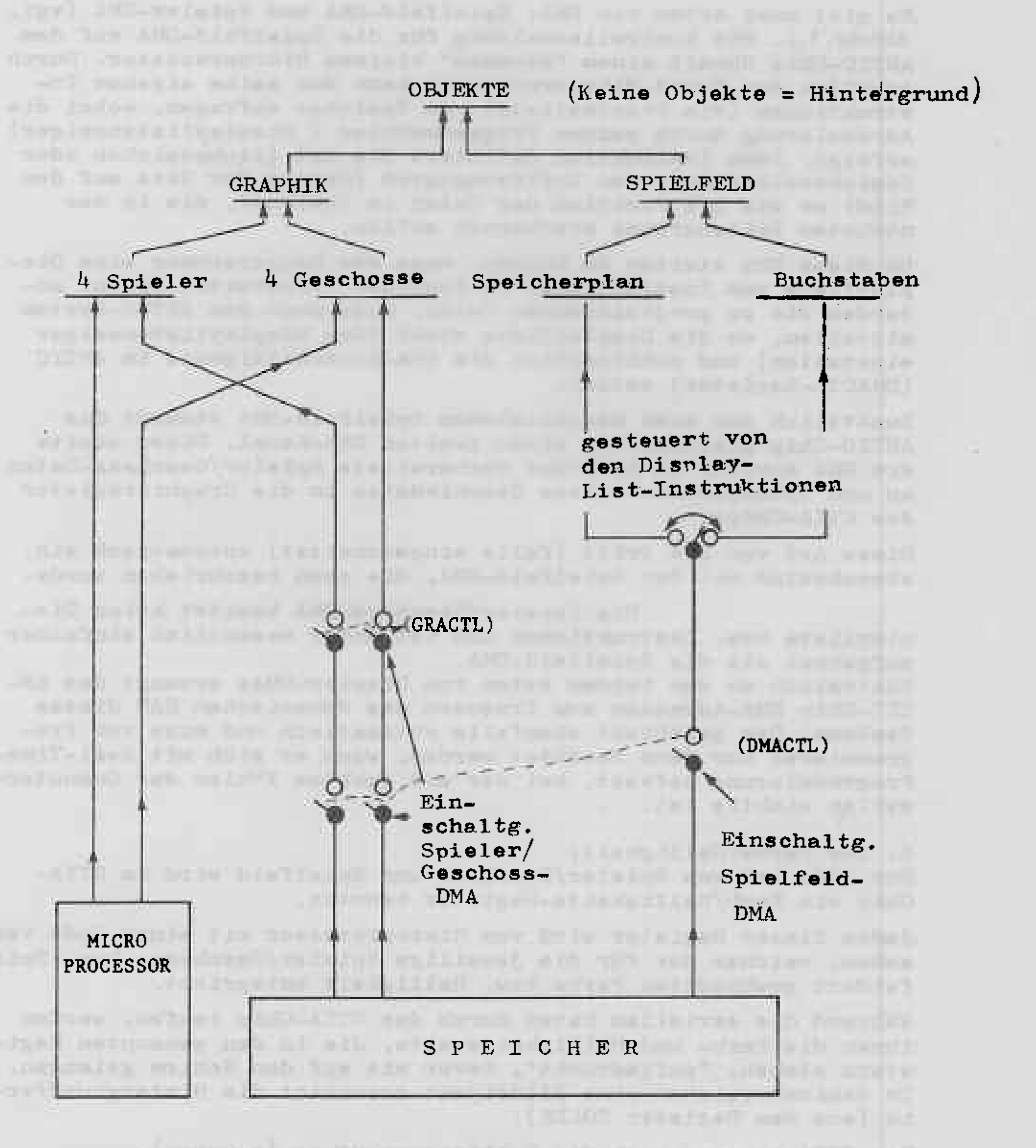


ABB.1: Quellen der Objekt-Projektion

Es gibt zwei Arten von DMA: Spielfeld-DMA und Spieler-DMA (vgl. Abbdg.1.). Die Kontrollschaltung für die Spielfeld-DMA auf dem ANTIC-Chip ähnelt einem "stummen" kleinen Mikroprozessor. Durch Anhalten des Haupt-Mikroprozessors kann der seine eigenen Instruktionen (die Displayliste) vom Speicher abfragen, wobei die Adressierung durch seinen Programmzähler (Displaylistenzeiger) erfolgt. Jede Instruktion definiert die Art (Alphazeichen oder Speicherplancode), den Auflösungsgrad (Grösse der Bits auf dem Bild) so wie die Position der Daten im Speicher, die in der nächsten Zeilengruppe erscheinen sollen.

Um diese DMA starten zu können, muss der Hauptrechner eine Displayliste von Instruktionen im Speicher vorbereitet haben, außerdem die zu projizierenden Daten, muss dann dem ANTIC-System mitteilen, wo die Displayliste steht (den Displaylistenzeiger einstellen) und schliesslich die DMA-Kontrollsignale im ANTIC (DMACTL-Register) setzen.

Zusätzlich zur oben beschriebenen Spielfeld-DMA steuert das ANTIC-Chip gleichzeitig einen zweiten DMA-Kanal. Diese zweite Art DMA spricht im Speicher vorbereitete Spieler/Geschoss-Daten an und transportiert diese Graphikdaten in die Graphikregister des CTIA-Chips.

Diese Art von DMA tritt (falls eingeschaltet) automatisch ein, abwechselnd mit der Spielfeld-DMA, die oben beschrieben wurde.

Die Spieler/Geschoss-DMA besitzt keine Displayliste bzw. Instruktionen und ist daher wesentlich einfacher aufgebaut als die Spielfeld-DMA.
Zusätzlich zu den beiden Arten von Display-DMAs erzeugt das ANTIC-Chip DMA-Adressen zum Erneuern des dynamischen RAM dieses Systems. Das geschieht ebenfalls automatisch und muss vom Programmierer nur dann beachtet werden, wenn er sich mit Real-Time-Programmierung befasst, bei der ein exaktes Zählen der Computerzyklen wichtig ist.

6. Zur Farbe/Helligkeit:
Für jede Art von Spieler/Geschoss und Spielfeld wird im CTIA-Chip ein Farb/Helligkeits-Register benutzt.

Jedes dieser Register wird vom Mikroprozessor mit einem Code versehen, welcher der für die jeweilige Spieler/Geschoss- bzw. Spielfeldart gewünschten Farbe bzw. Helligkeit entspricht.

Während die seriellen Daten durch das CTIA-Chip laufen, werden ihnen die Farb- und Helligkeitswerte, die in den genannten Registern stehen, "aufgedruckt", bevor sie auf den Schirm gelangen. In Schirmbereichen ohne Bildobjekt erscheint die Hintergrundfarbe (aus dem Register COLBK).

Das CTIA besorgt auch die Kollisionsprüfung (s.unten).

7.Zur Priorität:
Wenn bewegte Objekte wie Spieler und Geschosse sich gegenseitig bzw. das Spielfeld überlappen, muss entschieden werden, welches Element im Vordergrund stehen soll.

Man sagt von Objekten, die im Bild vor anderen stehen, dass sie gegenüber den anderen Priorität besitzen.

Die zugehörige Priorität wird allen Objekten vom CTIA-Chip zuge-

teilt, bevor sie seriell untereinander verbunden auf den Bildschirm gesandt werden. Die Priorität von Objekten kann vom Mikroprozessor durch Ansprechen des Kontrollregisters PRIOR kontrolliert werden. Die Funktion der Bits in diesem Register ist in der Tafel angegeben, die innerhalb der Beschreibung des PRIOR-Registers (vgl.weiter unten) abgebildet ist.

8. Zu Spielern und Geschossen:
Die Spieler und Geschosse sind kleine Objekte, die schnell in horizontaler Richtung bewegbar sind, indem man ihre Positionsregister ändert. Sie werden deshalb Spieler und Geschosse genannt, weil sie ursprünglich zum Gebrauch als Flugzeuge, Granaten, Spielfiguren usw. in Computerspielen entworfen worden waren.

Man kann sie jedoch auch für andere Zwecke benutzen. Die 4 Spieler/Geschoss-Farbregister ermöglichen, in Verbindung mit den 4 Spielfeldregistern und dem Register für die Hintergrundfarbe, die Projektion von 9 verschiedenen Farben zugleich.

Es gibt im Ganzen 4 Spieler und 4 Geschosse. Die 4 Geschosse können zu einem 5. Spieler zusammengefasst werden. Diese Objekte werden horizontal mit Hilfe von 8 Horizontal-Positionsregistern (HPOS (X)) aufgestellt. Positionsregister können durch den Prozessor jederzeit neu geladen werden, so dass sich ein Objekt viele Male längs einer TV-Zeile abbilden lässt.

Der Umriss eines Spieler/Geschoss-Elements wird durch die Daten in seinem Graphikregister (GRAF (X)) definiert. Spieler haben voneinander unabhängige 8-Bit-Graphikregister. Die 4 Geschosse besitzen 2-Bit-Register (innerhalb einer einzigen Adresse). Auch diese Register lassen sich jederzeit durch den Rechner neu laden, wobei die Änderung allerdings normalerweise während des Horizontal-Leertaktes erfolgt.

Die Daten jedes Graphikregisters werden jedesmal dann auf den Monitor projiziert, wenn der Horizontal-Synchronzähler den Stand des entsprechenden Horizontal-Positionsregisters erreicht. Es werden immer wieder dieselben Daten projiziert, solange die Graphikregister nicht mit neuen Daten geladen wurden.

Man kann die Spieler/Geschoss-Graphikregister entweder vom Rechner neu laden lassen (durch GRAF (X)) oder automatisch vom Speicher her über Direktzugriff (DMA) (vgl.Abb.2). Der Programmierer muss in diesem Fall die Objektgraphik im Speicher vorbereiten, die Spieler/Geschoss-Basisadresse (PMBASE) schreiben und die Spieler/Geschoss-DMA einschalten (DMACTL, GRACTL). Dann erfolgt die Übertragung der Objektgraphiken vom Speicher auf den Monitor vollautomatisch.

PMBASE beinhaltet das höchstwertige Byte (MSB) in der Adresse der Spieler/Geschoss-Graphik. Man bestimmt die Lage der Graphikdaten für das jeweilige Objekt durch Addition eines Abstandsbetrages zu PMBASE +256 (dezimal). Die Bytes, die zwischen der Basisadresse und den Geschossdaten liegen, werden von ANTIC nicht genutzt, so dass der Programmierer über sie verfügen kann.

Für einzeilige Auflösung werden nur die fünf höchsten Bits vom PMBASE benutzt; für zweizeilige die sechs höchsten Bits. Das be-

deutet, dass die Speicherung der Graphik an bestimmte Feldgrenzen gebunden ist. Zweizeilige Auflösung heisst, dass jedes Datenbyte für zwei Zeilen gleichermassen gilt (vgl. DMACTL, Bit 4).

64o (dezimale) Bytes (5 x 128) werden für eine zweizeilige Auflösung benötigt und 128o (5 x 256) für eine einzeilige.

Jedes Byte des Spielergraphik-Bereiches repräsentiert 8 Pixels, die auf den entsprechenden Zeilen des Monitors abgebildet werden sollen. Eine 1 zeigt an, dass der Farb/Helligkeitswert des Spielers in dem betreffenden Pixel erscheinen soll. Die Graphiken können jede beliebige Form haben, nicht nur Rechtecke wie in Abb.2.

Die Spielergraphiken können die gesamte Höhe des Schirms ausfüllen oder nur ein paar Zeilen hoch sein, wenn der Rest der Displaydaten nur aus Nullen besteht. Jedes Byte im Geschoss-Display entspricht ebenfalls 8 Pixels, nämlich zwei Pixels pro Geschoss. Die Pixelbreite kann 1, 2 oder 4 Colorclocks betragen. Sie wird von den SIZE-Registern bestimmt.

9. Zum Spielfeld:
Das Spielfeld wird immer durch die DMA erzeugt. Es gibt 4 Spielfelder, jeweils definiert durch ein eigenes Farbe/Helligkeits-Register und eine Kollisionsprüfung. Spielfelder werden durch 2 verschiedene DMA-Techniken aufgebaut: Speicherplan und Buchstaben-Zeichen. Beide Methoden sehen Instruktionslisten im Speicher vor, unabhängig von der Erzeugung von Spieler/Geschoss-Objekten.

Im Gegensatz zu Spielern und Geschossen gibt es für die Spielfelder keine Horizontal-Positionsregister. Jeder Spieler kann nur ein Bildbyte pro Zeile belegen. Ein Spielfeld dagegen kann bis zu 48 Bytes pro Zeile beanspruchen, da es ja unter Umständen den ganzen Bildraum ausfüllt.

Es gibt 3 verschiedene Spielfeldbreiten: Eng (128 Colorclocks), normal (16o Colorclocks) und breit (192 Colorclocks). Die Breite wird durch entsprechende Speicherung in das DMACTL bestimmt. Der Vorteil kleinerer Breite liegt darin, dass weniger Speicherraum besetzt und weniger Maschinenzyklen für die DMA "gestohlen" werden. Die OS-Graphik benutzt stets die normale Breite.

1o. Zur Display-Listung:
Die Display-Listung ist eine Folge von gespeicherten Display-Instruktionen. Diese Instruktionen sind entweder ein (1) oder drei (3) Bytes lang. Man kann die Displaylistung als ein Projektions-Programm betrachten und den Displaylistungs-Zähler, der diese Instruktionen abruft, kann man als Programmzähler für das Projektionsprogramm ansehen (1o Bits zur Zählung plus 6 Bits als Basis-Register).

Der Display-Listungszähler kann durch Einschreibung in DLISTH und DLISTL startbereit gemacht werden (entsprechend kann man die OS-Schattenregister SDLSTH bzw. SDLSTL nehmen). Dieser Zählerwert wird dann benutzt zur Adressierung der Displayliste, zum Abruf der Instruktion, zum Projizieren von 1 - 16 Datenzeilen auf den Monitor, zum Erhöhen des Displaylistenzählers, zum Abruf der nächsten Display-Instruktion usw., und zwar automatisch, ohne Steuerung durch den Rechner (vgl.DLISTL und DLISTH).

Die Spieler/Geschoss-Basis-Adresse = MSB (höchstes Bit) der Adresse.
Die Bildauflösung wird durch Bit 4 des DMACTL gesteuert.

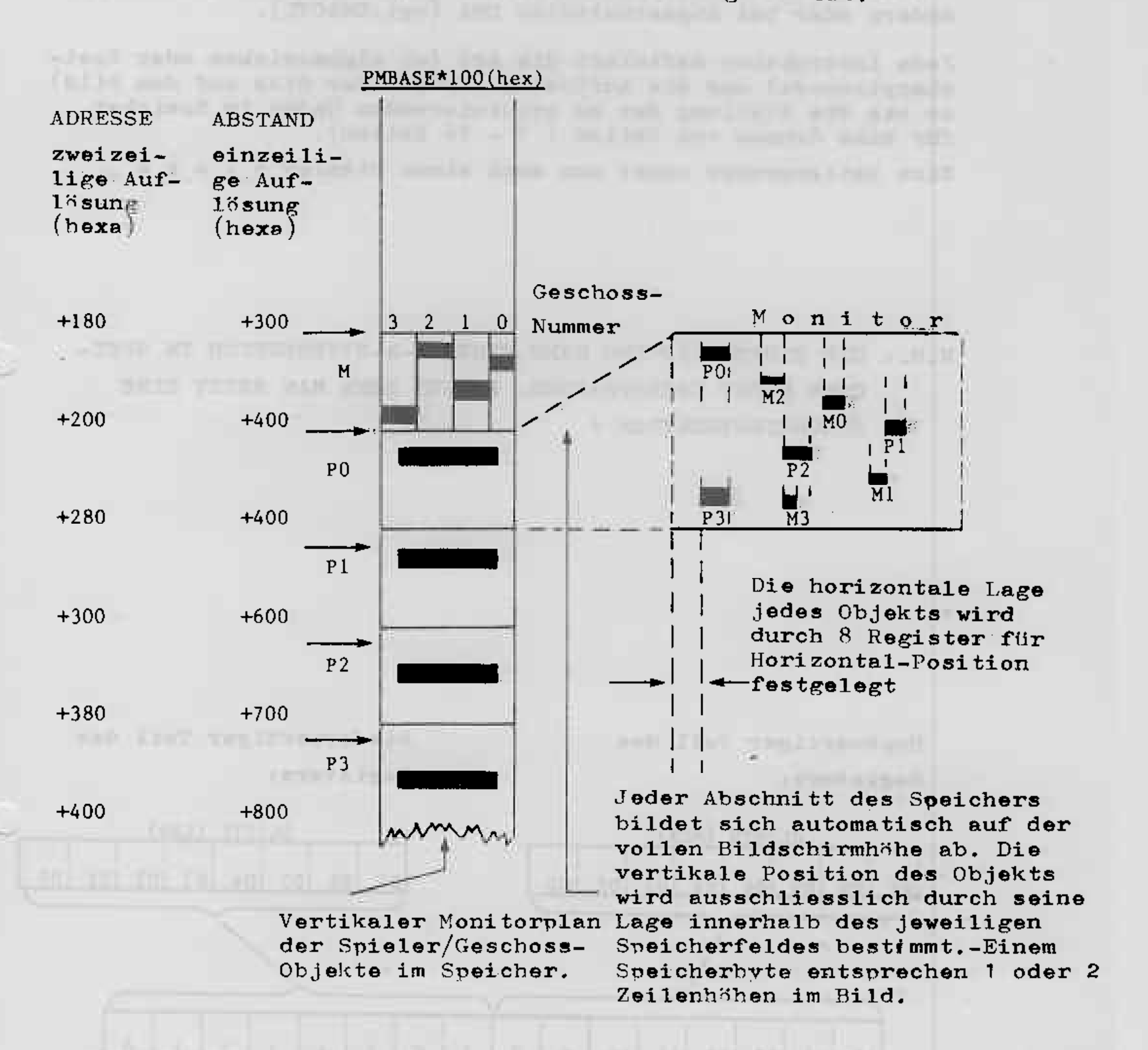


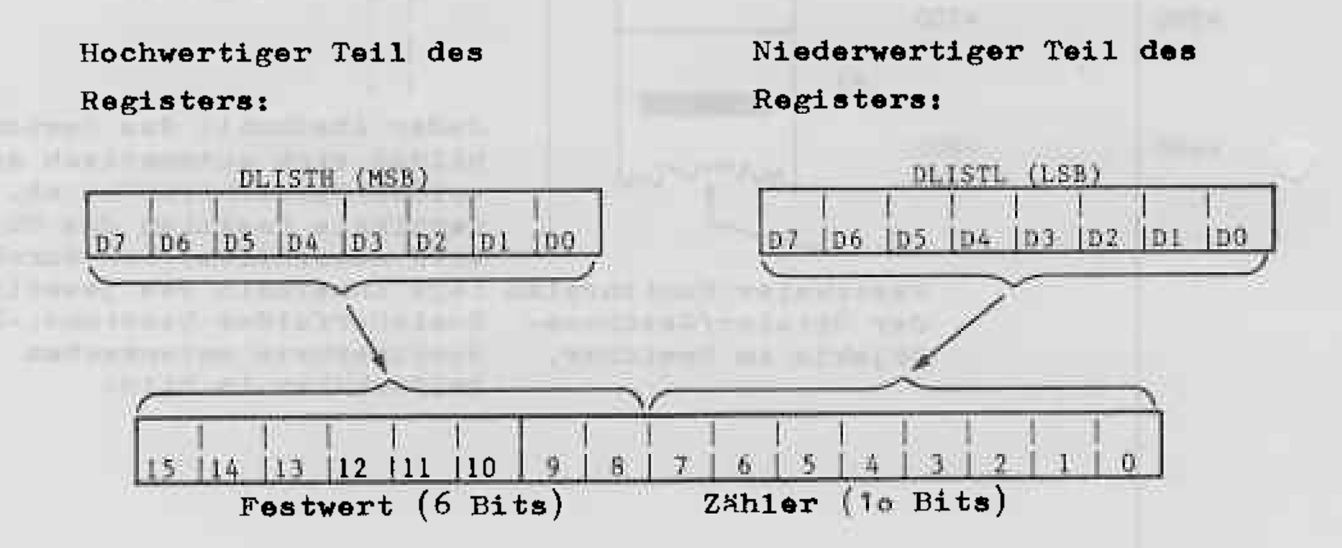
Abb. 2: Der Spieler-/ Geschoss - DMA.

DLISTL und DLISTH sollte man nur während Vertikal-Leertakten ändern oder bei abgeschalteter DMA (vgl.DMACTL).

Jede Instruktion definiert die Art (ob Alphazeichen oder Speicherplancode) und die Auflösung (Grösse der Bits auf dem Bild) so wie die Stellung der zu projizierenden Daten im Speicher für eine Gruppe von Zeilen (1 - 16 Zeilen).

Eine Zeilengruppe nennt man auch einen Display b l o c k .

N.B.: DIE DISPLAYLISTUNG KANN EINEN 1-K-BYTEBEREICH IM SPEICHER NICHT ÜBERSPRINGEN, ES SEI DENN MAN SETZT EINE SPRUNGINSTRUKTION !



DISPLAYLISTEN - ZÄHLER

HSCROL		XX		XX		XX		XX		XX		XX		XX		XX	Horizontal-Verschiebung
VSCROL			XX	XX			XX	XX			XX	XX			XX	XX	Vertikal-Verschiebung
LD MEM SCAN					XX	XX	XX	XX					XX	XX	XX	XX	Laden der Speicherstrecke (3 Bytes)
INST INTERRUPT									XX	XX	XX	XX	XX	XX	XX	XX	Unterbrechen d.Display-Instruktion
BLK 1	00								80								Leere eine Zeile
" 2	10								90								Leere 2 Zeilen
" 3-7																	Leere 3 - 7 Zeilen
" 8	70								F0								Leere 8 Zeilen
JMP	01								81								Sprung (Instruktion von 3 Bytes)
JVB	41								C1								Sprung u.Warten auf Vertikal-Leertakt (ebenfalls 3 Bytes lang)
CHR (40,2,8)	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2	Instruktionen für Schrift-Gänge
" (40,2,10)	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3	
" (40,4,8)	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4	
" (40,4,16)	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5	
" (20,5,8)	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6	
" (20,5,16)	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7	
MAP (40,4,8)	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8	Speicherplan-Instruktionen
" (80,2,4)	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9	
" (80,4,4)	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA	
" (160,2,2)	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB	
" (160,2,1)	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC	
" (160,4,2)	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD	
" (160,4,1)	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE	
" (320,2,1)	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF	

Anzahl von Bildzeilen pro Zeile
Anzahl der Farben (f.Hintergrund und Spielfeldarten)
Anzahl der horizontalen Zeilen (bei Standardbreite)

Abb. 3: Die OPCODES der Display - Instruktion.

Das Bit 7 einer Displaylist-Instruktion kann gesetzt werden, um Listungsunterbrechung zu erzeugen, falls Bit 7 des NMIEN gesetzt ist. Der Code für diese Unterbrechung kann die Farben oder Graphiken ändern, während die Mitte des Bildes erscheint. Die Art der Unterbrechung wird durch Abrfrage von NMIST ermittelt. NMIRES löscht das NMIST. Die laufende Operation des OS wird per Vektor über das VDSLST (hexadez.2oo und 2o1) zu der Displaylistungs-Unterbrechungsroutine des Benutzers geleitet.(Vgl.OS-Handbuch wegen Details dieser Programmierung.)

Bits 5 und 4 des Display-Art-Codes in einer Display-Instruktion werden benutzt, um vertikale und horizontale Bildverschiebung zu programmieren. Die Verschiebungsstrecke hängt ab von den Werten in den Registern VSCROL bzw. HSCROL (s.unten).

13. Zum Speicherstreckenzähler (Memory Scan Counter):
Der Programmierer kann mit diesem Zähler nicht direkt arbeiten. Das Register wird mit dem Wert geladen, der in den letzten beiden Bytes einer Instruktion (ausser bei Sprüngen) enthalten ist.

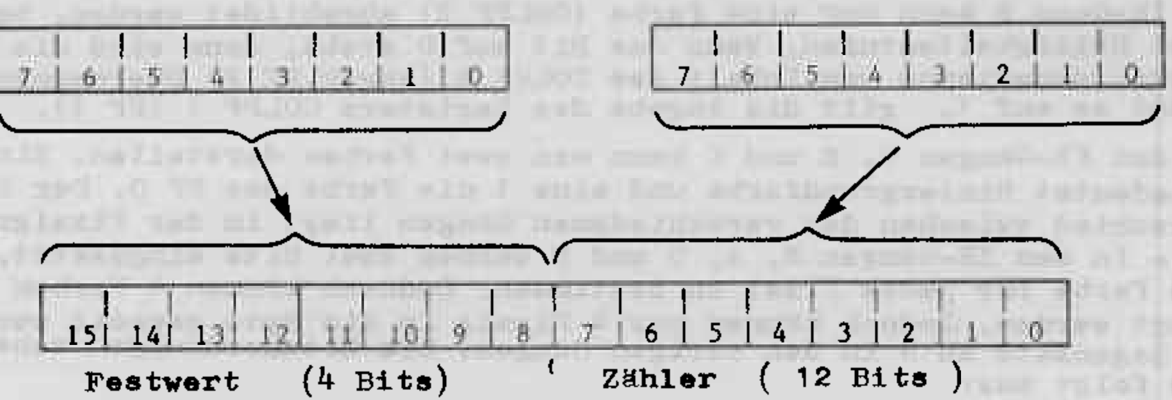
Der Speicherstreckenzähler zeigt auf die Adresse solcher Daten, die direkt projiziert werden (sogen. Speicherplan-Projektion) oder auf die Adressen von Buchstaben-Namen-Strings, die indirekt projiziert werden sollen (sogen.Buchstaben-Displays).

Eine 1 Byte grosse Instruktion lädt diesen Zähler nicht auf. Dies bedeutet einen kontinuierlichen Übergang im Speicher von dem zuvor projizierten zu den anschliessend zu projizierenden Daten.

Da der Streckenzähler in Wirklichkeit aus 4 Registerbits und 12 eigentlichen Zählbits besteht, kann ein zusammenhängender Speicherbereich die Grenzen von jeweils 4 K Bytes nicht überschreiten, es sei denn, der Streckenzähler wird mit einer 3 Byte großen Ladeinstruktion (Load Memory Scan Counter) neu eingestellt.

Höchstwertiges Byte (3.Byte einer 3-Byte-Instruktion):

Niedrigstwertiges Byte (2.Byte einer 3-Byte-Instruktion):



14. Zu den Speicherplan-Displayinstruktionen:
Daten im Speicher werden (per Adressierung über den Speicherstreckenzähler) durch Ausführung einer Speicher-(Bit-)Plan-Displayinstruktion direkt projiziert. Gleichzeitig mit der Projektion werden die Daten in ein Verschieberegister gebracht, so dass sie für soviele Zeilen des Monitors wiederholt werden können, wie es die Instruktion verlangt.

Speicherstrecken-Zähler;
adressiert jedes Byte

Speicher

Der einer Zeile entsprechende
Speicherinhalt wird in das
Verschieberegister geladen

Verschiebe-
Register

Die Daten im Verschieberegister werden über vier Bildzeilen hin projiziert.

In den Displaygängen 8 bis F des Instruktionsregisters (IR) werden ein oder zwei Bits benutzt, um zu definieren, was auf jedes einzelne Pixel projiziert werden soll. Die Pixelgrössen reichen von 1/2 Clock x einer Zeilenhöhe bis zu 4 Clocks x 8 Zeilenhöhen.

Die meisten dieser Graphikgänge werden über das OS und BASIC gesteuert (BASIC GRAPHICS - Befehl). Zwei Gänge, C und E, werden nicht vom OS geleitet. Diese Gänge besitzen rechteckige Pixels, die etwa zweimal so breit wie hoch sind.

Im IR-Gang F kann nur eine Farbe (COLPF 2) abgebildet werden, bei zwei Helligkeitsstufen. Wenn das Bit auf 0 steht, dann wird die Helligkeitsanweisung vom Inhalt des COLPF 2 (abgek.PF 2) übernommen; steht es auf 1, gilt die Angabe des Registers COLPF 1 (PF 1).

In den IR-Gängen 9, B und C kann man zwei Farben darstellen. Eine 0 bedeutet Hintergrundfarbe und eine 1 die Farbe des PF 0. Der Unterschied zwischen den verschiedenen Gängen liegt in der Pixelgröße.- In den IR-Gängen 8, A, D und E werden zwei Bits eingesetzt, um die Farbe für jedes Pixel zu bestimmen. Dadurch können 4 Farben erzeugt werden. Jedoch können nur 4 Pixels in ein Byte gepackt werden im Gegensatz zu 8 in den vorigen Gängen. Die Bitanordnungen sehen wie folgt aus:

VERSCHIEBEREGISTER	7 6	5 4	3 2	1 0	7 6	5 4	3 2	1 0

2 Bits bilden 1 Pixel

Speicherplan-Displaygänge

(Memory Map Display Modes) :

OS- und BASIC-Gänge	Instr. Reg. HEX	Colors per Gang	Pixels per Normal-Zeile	Bytes per Norm.-Zeile	Zeilen per Pixel	Color Clocks per Pixel	Bits per Pixel	Bit-Werte in Pixel	Farb-Reg.-Wahl
3	8	4	40	10	8	4	2	00 01 10 11	BAK PF0 PF1 PF2
4	9	2	80	10	4	2	1	0 1	BAK PF0
5	A	4	80	20	4	2	2	00 01 10 11	BAK PF0 PF1 PF2
6	B	2	160	20	2	1	1	0 1	BAK PF0
–	C	2	160	20	1	1	1	0 1	BAK PF0
7	D	4	160	40	2	1	2	00 01 10 11	BAK PF0 PF1 PF2
–	E	4	160	40	1	1	2	00 01 10 11	BAK PF0 PF1 PF2
8	F	1 ½	320	40	1	½	1	0 1	PF2 PF1 (LUM)

15. Zu den Zeichen-Display-Instruktionen:
Der erste Schritt bei der Benutzung des Zeichenplan-Gangs besteht darin, einen Zeichensatz (Character set) im Speicher aufzubauen (es sei denn, der eingebaute Satz des OS auf Adresse hexadez. E000 soll verwandt werden). Der Zeichensatz enthält 8 Bytes mit Daten für die Gestaltung jedes Grosszeichens. Die Bedeutung der Daten hängt vom jeweiligen Gang ab. Der Satz kann 64 oder 128 Zeichen enthalten, was ebenfalls vom Gang abhängt.

Das MSB (höchstwertige Byte) der Adresse des Zeichensatzes wird im CHBASE (oder dem Schatten-CHBAS des OS) abgestellt. Nur die höchsten 6 oder 7 Bits des CHBAS (bzw. des Schatten-CHBAS) werden benutzt (vgl.CHBASE-Beschreibung im nächsten Teil dieser Abhandlg.). Die anderen ein oder zwei Bits und die unteren Bytes dieser Adresse werden als auf Null stehend angenommen, so dass der Zeichensatz bei einer passenden Speichergrenze beginnen muss.

Der nächste Schritt besteht im Aufstellen der Displaylistung für den gewünschten Gang. Dann wird das eigentliche Display konstruiert. es setzt sich zusammen aus einem Zeichenstring, aus Namen oder aus Codes. Jeder Name nimmt ein Byte ein. Die letzten 6 oder 7 Bits des Namens wählen Zeichen aus. Bei einem Satz von 64 Zeichen würde der Name von 0 - 63 (dezimal) reichen. Für einen 128er Satz wäre das der Bereich von 0 - 127 (dezimal). Das obere der beiden Bits des Namen-Bytes dient der Spezifizierung der Farbe oder einer anderen speziellen Information, je nach gewähltem Gang.

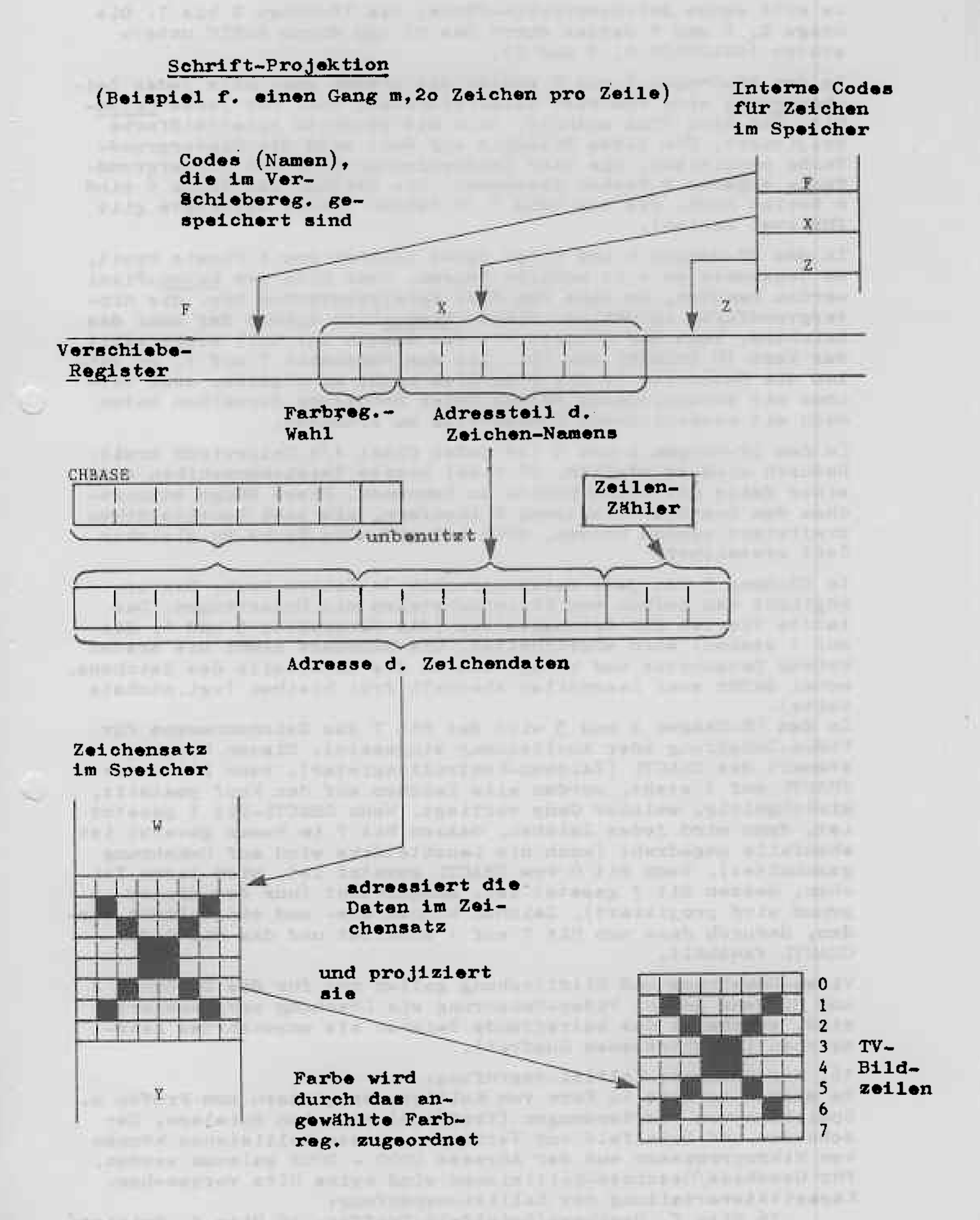
Zeichennamen (Codes) werden durch den Speicherstreckenzähler abgerufen und dann in Verschieberegister abgestellt. Bei jeder Projektionszeile rückt das Verschieberegister seinen Inhalt um eine Stelle weiter, wobei jedoch nur der Namensteil der Zeichenadresse bewegt wird (s. Tabelle unten).

Nach Projizierung einer vollen Zeile von Zeichendaten wird der Zeichenzähler um 1 erhöht. Die nächste Zeile spricht wieder für sich alle Zeichen über den Namen an.

In Gängen mit 2o Zeichen pro Zeile werden die 7 höchsten Bits des CHBASE benutzt. Das setzt voraus, dass der Zeichensatz bei einer 512-Byte-Grenze anfängt. Der Satz muss 64 Zeichen zu je 8 Bytes enthalten, was eine Summe von 512 Bytes pro Satz ergibt.

Die Gänge von 4o Zeichen pro Zeile verwenden die 6 höchsten Bits des CHBASE, so dass der Zeichensatz bei einer 1-K-Byte-Grenze beginnen muss. Der Satz muss 128 Zeichen mit 8 Bytes pro Zeichen enthalten. Das ergibt 1o24 Bytes.

Hexa-Code	Graphik-gang	Zeichen pro Zeile	Anzahl d. Farben	Bytes per Buchstab.	Anz.d. Buchst. pro Satz	Bytes pro Satz
2	0	40	2	8	128	1024
3	-	40	2	8	128	1024
4	-	40	4	8	128	1024
5	-	40	4	8	128	1024
6	1	20	5	8	64	512
7	2	20	5	8	64	512



Schrift-Projektion
(Beispiel f. einen Gang m,2o Zeichen pro Zeile)
Interne Codes
für Zeichen
im Speicher
Codes (Namen),
die im Ver-
Schiebereg. ge-
speichert sind
F
X
Z
Verschiebe-
Register
Farbreg.-
Wahl
Adressteil d.
Zeichen-Namens
CHBASE
Zeilen-
Zähler
unbenutzt
Adresse d. Zeichendaten
Zeichensatz
im Speicher
W
Y
adressiert die
Daten im Zei-
chensatz
und projiziert
sie
Farbe wird
durch das an-
gewählte Farb-
reg. zugeordnet
0
1
2
3
4
5
6
7
TV-
Bild-
zeilen

Es gibt sechs Zeichengraphik-Gänge, die IR-Gänge 2 bis 7. Die Gänge 2, 6 und 7 werden durch das OS und durch BASIC unterstützt (GRAPHICS 0, 1 und 2).

In den IR-Gängen 6 und 7 wählen die oberen zwei Bits jedes Zeichennamens eine von vier Spielfeldfarben aus. Für jedes Daten-Bit, das eine Eins enthält, wird die gewählte Spielfeldfarbe projiziert. Für jedes Datenbit auf Null wird die Hintergrundfarbe projiziert. Die vier Zeichenfarben plus der Hintergrundfarbe ergeben 5 Farben insgesamt. Die Zeichen des Gangs 6 sind 8 Zeilen hoch, die von Gang 7 16 Zeilen (jedes Datenbyte gilt für zwei Zeilen).

In den IR-Gängen 4 und 5 ist jedes Zeichen nur 4 Pixels breit, im Gegensatz zu 8 in anderen Gängen. Zwei Bits per Daten-Pixel werden benutzt, um eine von drei Spielfeldfarben bzw. die Hintergrundfarbe zu wählen. Sieben Namensbits dienen der Wahl des Zeichens. Wenn das höchste Bit des Namens auf Null steht,wählt der Wert 10 (binär) das PF1. Ist das Namensbit 7 auf 1, so wählen die Datenbits 1o das PF2. Dies macht es möglich, zwei Zeichen mit verschiedenen Farben unter Benutzung derselben Daten doch mit verschiedenen Namensbytes zu erzeugen.

In den IR-Gängen 2 und 3 ist jedes Pixel 1/2 Colorclock breit. Dadurch wird es möglich, 48 Pixel breite Zeichengraphiken auf einer Zeile mit Normalbreite zu bekommen. Diese Gänge entsprechen dem Speicherdirektgang F insofern, als zwei Leuchtstärken projiziert werden können, doch ist nur eine Farbe zu gleicher Zeit erstellbar.

Im IR-Gang 3 ist jede Zeichengraphik 1o Zeilen hoch. Das ermöglicht den Aufbau von Kleinbuchstaben mit Unterlängen. Das letzte Viertel des Zeichensatzes (die Namensbits 5 und 6, die auf 1 stehen) wird abgeschaltet. Die Hardware nimmt die ersten beiden Datenbytes und bewegt sie an die Unterseite des Zeichens, wobei dafür zwei Leerzeilen oberhalb frei bleiben (vgl.nächste Seite).
In den IR-Gängen 2 und 3 wird das Bit 7 des Zeichennamens für Video-Umkehrung oder Auslöschung eingesetzt. Diesen Vorgang steuert das CHACTL (Zeichen-Kontrollregister). Wenn Bit 2 des CHACTL auf 1 steht, werden alle Zeichen auf den Kopf gestellt, gleichgültig, welcher Gang vorliegt. Wenn CHACTL-Bit 1 gesetzt ist, dann wird jedes Zeichen, dessen Bit 7 im Namen gesetzt ist, ebenfalls umgedreht (auch die Leuchtstärke wird auf Umkehrung geschaltet). Wenn Bit 0 vom CHACTL gesetzt ist, wird jedes Zeichen, dessen Bit 7 gesetzt ist, ausgelöscht (nur der Hintergrund wird projiziert). Zeichen können aus- und eingeblinkt werden, dadurch dass man Bit 7 auf 1 schaltet und das Bit 0 des CHACTL festhält.

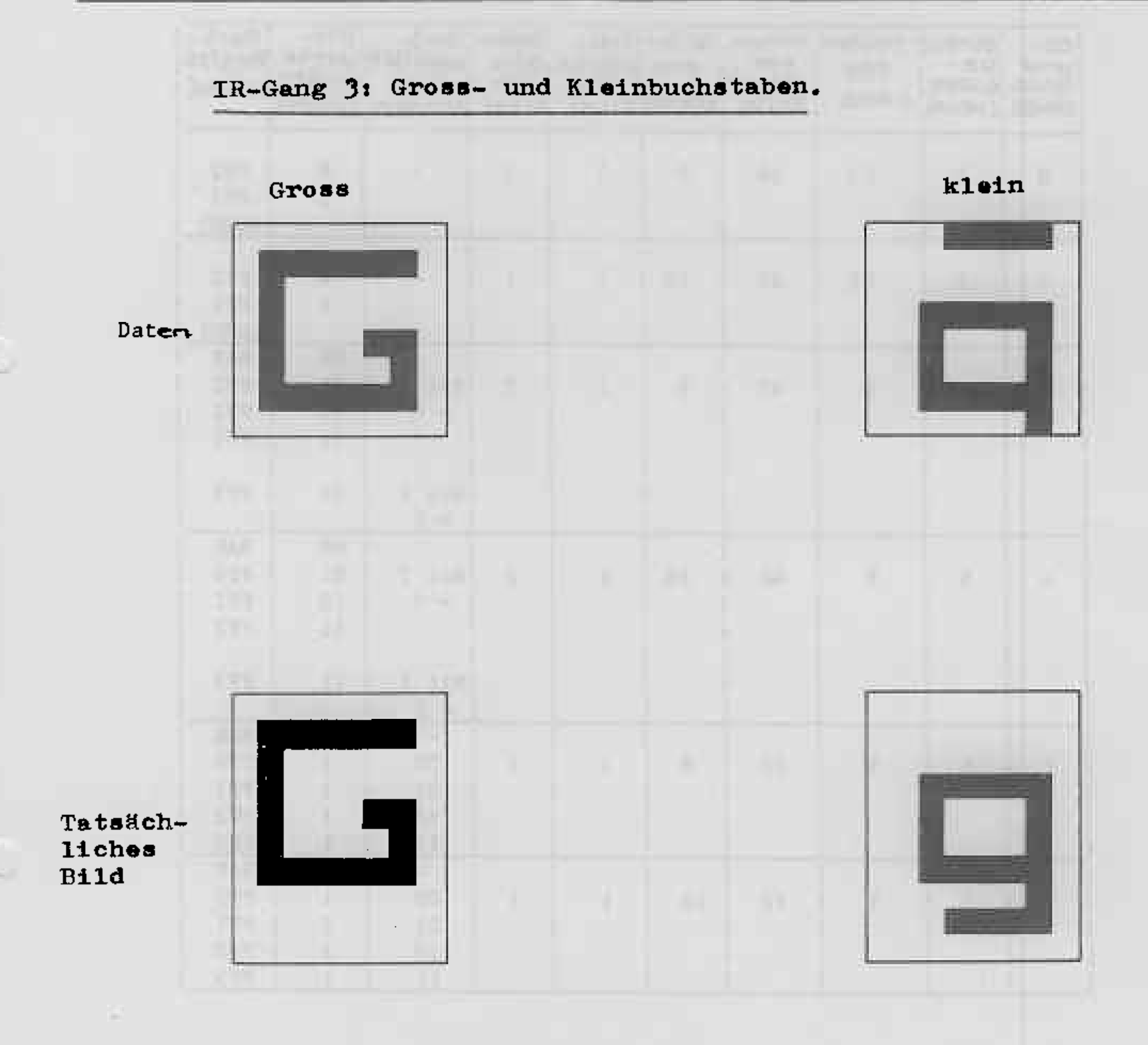
Video-Umkehrung und Bildlöschung gelten nur für die IR-Gänge 2 und 3. Wenn sowohl Video-Umkehrung wie Löschung programmiert sind, erscheint das betreffende Zeichen als umgekehrtes Leerzeichen (geschlossenes Quadrat).

16. Zur Hardware-Kollisionsprüfung:
Es stehen 6o Bits in Form von Kollisionsregistern zum Prüfen u. Speichern von Überlappungen (Treffern) zwischen Spielern, Geschossen und Spielfeld zur Verfügung. Diese Kollisionen können vom Mikroprozessor aus der Adresse D000 - D00F gelesen werden. Für Geschoss/Geschoss-Kollisionen sind keine Bits vorgesehen.
Kapazitätsverteilung der Kollisionsprüfung:

16 Bits f. Geschoss/Spielfeld-Treffer, 16 Bits f. Spieler/Spielfeld-Treffer, 16 Bits für Geschoss/Spieler-Treffer, 12 Bits für Spieler/Spieler-Treffer. (P0/P0 usw. wird jeweils als 0 gespeichert)

Der Speicherplan-Direktgang mit 1/2 Clock Auflösung (IR Code 1111) und der Zeichengang von 1/2 Clock Auflösung (IR-Codes 0011 und 0010) gelten beide als Spielfeld-Kollisionen des Typs 2 und werden im Bit 2 der jeweiligen Spielfeld-Kollisionsregister gespeichert.

IR-Gang 3: Gross- und Kleinbuchstaben.



Projektions-Gänge
mit Zeichen-Plan

OS- und BASIC-GÄNGE	BEFEHLS-REGISTER (HEXA)	FARBEN PRO GANG	Zeichen pro Normal-Zeile	Zeilen pro Zeichen	Color-Clocks pro Pixel	Daten-Bits pro Pixel	Farb-wahlbits im Namen	Bit-werte in den Daten	Farb-Regist.-Wahl
0	2	1½	40	8	½	1	–	0 1	PF2 PF1 (LUM)
–	3	1½	40	10	½	1	–	0 1	PF2 PF1 (LUM)
–	4	5	40	8	1	2	Bit 7 = 0 Bit 7 = 1	00 01 10 11 11	BAK PF0 PF1 PF2 PF3
–	3	5	40	16	1	2	Bit 7 = 0 Bit 7 = 1	00 01 10 11 11	BAK PF0 PF1 PF2 PF3
1	6	5	20	8	1	1	– 00 01 10 11	0 1 1 1 1	BAK PF0 PF1 PF2 PF3
2	7	5	20	16	1	1	– 00 01 10 11	0 1 1 1 1	BAK PF0 PF1 PF2 PF3

17. Vertikaler und Horizontaler Feinvorschub:
Es ist schwierig, Spielfeldobjekte ruckfrei zu verschieben. Ein Speicherplan-Spielfeld kann man gut dadurch bewegen, dass man einzelne Speicherabschnitte neu lädt. Dies Verfahren verbraucht aber sehr viel Zeit, wenn grosse Bildflächen so verschoben werden. Objekte auf einem Zeichen-Spielfeld lassen sich leicht, aber nur verwackelt, durch Änderung des Zeilenzählers bewegen. Das ergibt allerdings einen Positionssprung von einem Zeichen zum nächsten, keinen stufenlosen Übergang.

Wegen dieser Schwierigkeiten stehen die Hardwareregister VSCROL und HSCROL so wie Zähler zur Verfügung, durch die ein horizontaler bzw. vertikaler Feinvorschub möglich wird, und zwar bis zu jeweils einer Zeichenbreite waagerecht bzw. einer Zeichenhöhe senkrecht. Wenn diese Strecke durch Erhöhen der Reg.-Werte feinverschoben ist, wird der Speicher mit neuen Daten geladen(die Speicherwerte werden angepasst) od. die Zeilenzähler verändert, so dass die Feinbewegung um eine neue Zeichendistanz weiterlaufen kann.

Vertikaler Vorschub: Eine Spielfeldzone auf dem Monitor kann durch VSCROL und Bit 5 der Displaylist-Instruktion nach oben verschoben werden. Die Displayblocks an den unteren und oberen Grenzen der Zone müssen eine variable vertikale Grösse besitzen; insbesondere der erste Displayblock innerhalb der Zone muss von oben her gekürzt werden, während der letzte von unterhalb gekürzt werden muss (d.h., dass nicht alle oberen bzw. unteren Blocks projiziert werden).

Die vertikale Dimension wird von einem 4-Bit-Zähler im ANTIC gesteuert, der sich "Deltazähler" (DCTR) nennt. Ohne vertikalen Vorschub beginnt dieser Zähler bei Null auf der ersten Zeile und zählt dann aufwärts bis zu einem Normalwert, der durch die laufende Displayinstruktion bestimmt wird. Z.B. ist für Gross- und Kleinbuchstaben-Textprojektion der Endwert 9, für 5-farbige Zeichen-Projektionen ist er 7 oder 15.

Wenn Bit 5 der Instruktion zwischen aufeinanderfolgenden Projektionsblocks unverändert bleibt, wird der zweite Block in normaler Form projiziert. Wenn das Bit 5 der Instruktion zwischen zwei zusammenhängenden Blocks auf Null schaltet, dann beginnt der zweite Block mit Delta=0, wie gewöhnlich, zählt jedoch aufwärts bis Delta=VSCROL, statt den Normalwert zu nehmen. Dadurch wird dieser Displayblock von unten her gekürzt.

Die direkte Methode um eine vertikal zu verschiebende Zone zu definieren, besteht darin, Bit 5 in der ersten Displayinstruktion für diese Zone auf 1 zu setzen und ebenso in den folgenden Blocks ausser dem letzten. Wenn das VSCROL nicht während des Ablaufs verändert wird, so erbigt sich eine vollständig sich verschiebende Zone mit einer feststehenden Zeichenzahl (vorausgesetzt, der Wert in VSCROL überschreitet nicht die jeweilige Blockgrösse). Wenn N die normale Blockgrösse ist, so hat der obere Block N-VSCROL Zeilen (N>VSCROL) und der letzte Block hat VSCROL+1 Zeilen: (N - VSCROL)+(VSCROL+1)=N+1.

Im folgenden wird das Beispiel einer Vorschubzone, oberer Block, mit 8 VSCROL-Werten für N = 8 gezeigt.
Horizontaler Vorschub wird unter HSCROL im nächsten Teil beschrieben.

Einfaches Beispiel für Displaylistung:

BASIC beginnt im OS-Graphikgang 0, der 4o Zeichen breit und 24 Zeilen hoch projiziert. Das entspricht dem IR-Gang 2 mit normaler Bildbreite. Das OS baut die Displayliste nahe der RAM-Obergrenze auf mit einer Platzreserve für die Zeichennamen an der Obergrenze des RAM. In einer 32-K-Byte-Anlage wird die Displayliste bei hexadezimal 7C2o beginnen. Die nächsten drei Bytes sind die von hexadezimal 70, die 24 Leerzeilen erzeugen. Das nächste Byte ist eine hexadezimale 42. Die 4 weist die Maschine an, den Zeilenzähler mit der nachfolgenden Adresse zu laden (7C4o). Dies ist die Adresse der abzubildenden Daten. Die 2 weist die Maschine an, eine Zeile von Zeichen in IR-Gang 2 zu projizieren. Die folgenden 23 Bytes spezifizieren weitere 23 Zeilen mit Gang-2-Zeichen. Hexadezimal 41 ist der Code für Springen und Warten, bis das Ende des nächsten Vertikal-Leertaktes erreicht ist. Die nächsten 970 Bytes stellen die Liste der abzubildenden Daten dar, jeweils 4o Bytes pro Zeile. Das OS muss den Zeiger für die Displayliste (DLISTH und DLISTL) auf die Startadresse der Displayliste einstellen (7C2o). Es setzt außerdem CHBASE auf das MSB (höchstwertiges Byte) der Zeichensatz-Adresse (E0).

Dies Beispiel ist als einfach zu bezeichnen, weil nur ein Gang benutzt wird und der Zeilenzähler nur an einem Punkt geladen wird. Es ist möglich, verschiedene Gänge auf verschiedenen Zeilen zu programmieren, Zeichensätze und Farben zu ändern usw. (s.Teil IV)

Displayliste im OS - Gang 0 (4o Zeichen X 24 Zeilen)

Addresse(hexa)	Data (hexa)	
7C20	70	24 Leerzeilen
	70	
	70	
	42	Lade Zeilenzähler neu mit 7C40, IR - Gang 2
	40	
	7C	
	2	23 weitere Instruktionen in IR - Gang 2
	2	
	2	
	.	
	.	
	.	
	2	
	2	
	2	
	41	Springe zurück auf 7C20 und warte auf das Ende des Vertikal-Leertaktes
	20	
	7C	
7C40		96o Bytes Bilddaten (Zeichennamen)

18. Zur Zykluszählung:
Wie oben bereits erklärt, läuft der 65o2 Mikroprozessor der ATARI 8oo mit einer Frequenz von 114 Maschinenzyklen pro Bildzeile (1,79 MHZ). Ein Bilddurchlauf (PAL) hat 31o Zeilen, in einer Sekunde erscheinen 5o Bilder.

Jeder Maschinenzyklus entspricht einem Zeilenstück von zwei Color Clocks. Eine ganze Zeile hat 228 Color Clocks. Die Graphikgänge mit der höchsten Auflösung haben eine Pixelgrösse von 1/2 Col.Cl. X einer Zeilenhöhe. Der Horizontal-Leertakt ist 4o Maschinenzyklen lang. Dieser Takt ist der Rücksprung des Bildstrahls zum linken Bildrand, wo er eine neue Zeile beginnt.

Ein Befehl "Warte auf Synchronisation" (WSYNC) soppt den 65o2. Er läuft genau 7 Maschinenzyklen vor dem Anfang der neuen Bildzeile weiter. Das Programm kann also Graphen bzw. Farben während des Horizontal-Leertaktes für die folgende Zeile ändern.

Der ANTIC-Chip nimmt dem 65o2 einige Zyklen weg, um den Speicher neu einzurichten und ggflls. Daten abzugreifen. Als allgemeine Regel muss man sich merken, dass jeder Datenabgriff einen Maschinentakt beansprucht. Erstreckt sich eine Speicherplan-Bildinstruktion über mehrere Zeilen, so werden deren Daten nur auf der ersten Zeile abgegriffen. Speichererneuerung nimmt 9 Takte pro Zeile in Anspruch, wenn sie nicht durch einen Graphikgang mit hoher Auflösung vorweggenommen wird. Während des Vertikal-Leertaktes hält die Speichererneuerung an.

Geschoss-DMA verbraucht einen Takt pro Zeile im Gang mit einzeiliger Auflösung und einen Takt jede zweite Zeile bei einem Gang mit zweizeiliger Auflösung. Geschoss-DMA kann auch ohne Spieler-DMA angeschaltet werden, nicht aber umgekehrt (vgl.DMACTL- und GRACTL-Bits). Die Spieler-DMA erfordert 4 Takte pro Zeile bzw.Doppelzeile,

je nach benutzter Auflösung.

Jeder Abruf eines Displaylisten-Bytes verbraucht einen Maschinen-Takt, so dass drei Takte für eine Instruktion von drei Bytes verlorengehen. Die direkte Speicheradressierung (DMA) beim Abruf von Spieler/Geschoss- und Displaylist-Instruktionen erfolgt während des Horizontal-Leertaktes, falls sie für die nächste Zeile benötigt wird.

In Speicherplan-Gängen werden die Graphikdaten nach Bedarf während des Ablaufs der ersten Zeile der Programmierten Displaylisten-Instruktion abgerufen und dann durch ANTIC für weitere Verwendung gespeichert. In Zeichengängen werden die Zeichencodes während des Ablaufs der ersten Zeile jeder Zeichenreihe zusammen mit den Graphikdaten für diese Zeile abgerufen. Während der nächsten Zeilen werden nur noch die Graphikdaten geholt, da ANTIC die Zeichendaten festhält.
Im 4o x 24 - Zeichengang, bei normaler Monitorbreite, werden die meisten Arbeitstakte während der oberen Zeile jeder Zeichenreihe benötigt, um die Zeichencodes und zugehörigen Daten abzufragen. Deshalb bleibt nur noch für einen Takt Speichererneuerung Zeit, statt der üblichen neun.

Bei schmalem Bild ist weniger DMA-Aufwand nötig, so dass dann zwei Takte für Speichererneuerung frei sind. Die Speichererneuerung geschieht schnell genug, um die verlorenen Takte in den Gängen mit hoher Auflösung zeitmässig auszugleichen. Sobald Speichererneuerung auf einer Zeile einsetzt, schaltet sie sich alle 4 Takte neu ein, es sei denn dass sie durch eine DMA gestört wird.

Alle Unterbrechungen erreichen den 65o2 gegen Endes des Horizontal-Leertaktes. Bei normalen oder schmalen Monitoren beginnt die Speicherauffrischung nach dem Ende des Horizontal-Leertaktes.

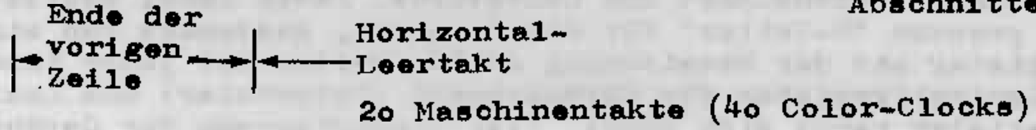
Die Zeit, in der ANTIC Takte entnimmt, kann zwar bestimmt werden, doch hängt sie vom benutzten Graphikgang, von der Bildbreite und davon ab, ob horizontaler Vorschub eingeschaltet ist oder nicht. Letzteres erferdert extra Graphikdaten (s. HSCROL).

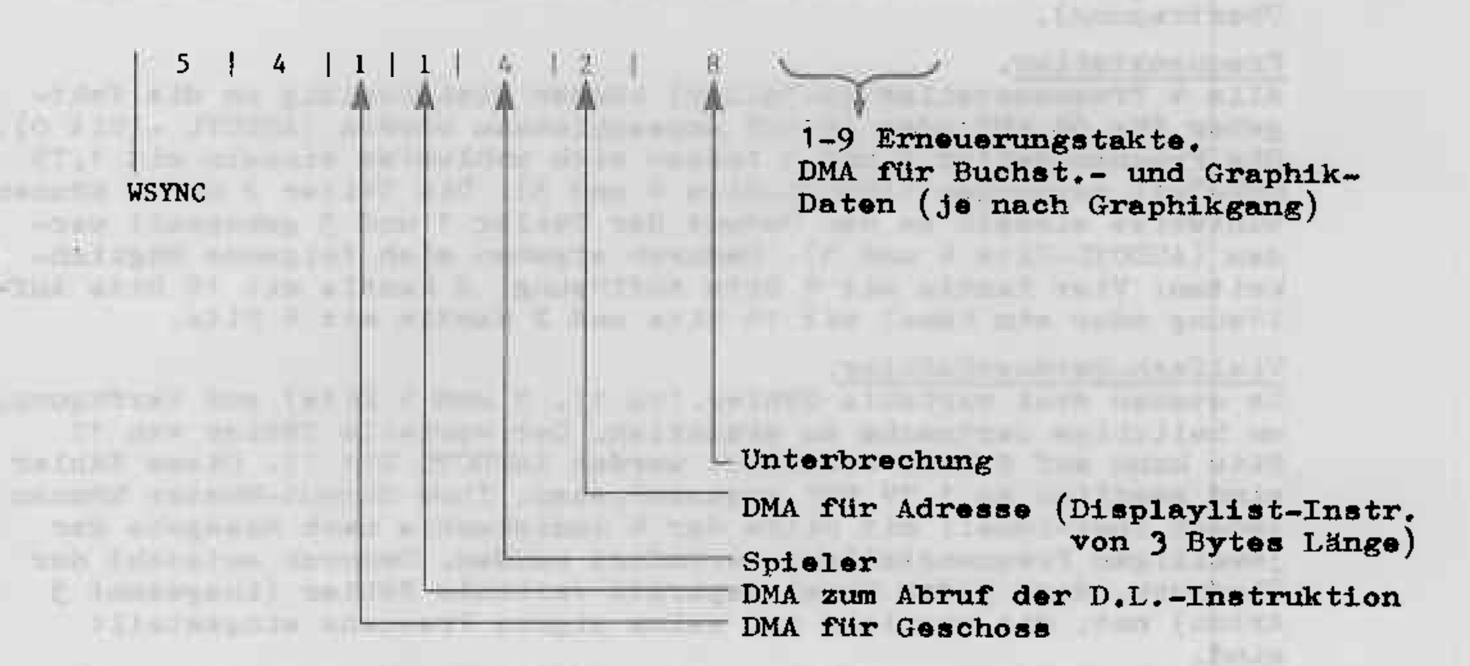
ANTIC bewirkt horizontalen Vorschub um eine gerade Anzahl von Colorclocks durch Verzögern des Zeitpunktes, zu dem es die Daten direkt abrufen lässt. Damit auch Bewegungen von einem Colorclock möglich sind (was einem halben Maschinenzyklus entspricht), besitzt ANTIC eine innere Verzögerung von 1 Colorclock.

Theoretisch ist es möglich, ein Programm so zu schreiben, dass Graphen oder Farben "im Fluge" geändert werden, d.h. mitten in einer TV-Zeile. Da jedoch eine grosse Anzahl von DMAs zwischendurch ablaufen, wird das Zählen der Takte oft so kompliziert, dass im Rahmen dieser Beschreibung darauf nicht eingegangen werden kann.
Es gibt eine Anzahl von Verzögerungen in Verbindung mit der Projektion von Graphiken. Sie treten im ANTIC und im CTIA ein. ANTIC sendet Daten an das CTIA, welches die Farbinformation hinzufügt. Deshalb ist die Zeiteinteilung für das Ändern von Farben während des Zeilenablaufs anders als für das Ändern von Graphik im "Fluge".

Horizontal-Leertakt-DMA

Bei eingeschaltetem DMA werden Maschinentakte in den unten angegebenen Abschnitten weggenommen.





Beispiel für Zählung von Takten:
Dies Beispiel benutzt die Displaylistung von 4o Zeichen x 24 Zeilen, die zwei Seiten weiter gezeigt wird. Die Listung ist 32 Bytes lang, so dass ihre DMA 32 Maschinentakte verbraucht. Die DMA für die Zeichennamen nimmt 96o Takte, die für die Zeichendaten 8 x 96o in Anspruch. Die Speichererneuerung verbraucht 9 Takte pro Zeile, ausser für die 24 Zeilen, auf denen die Zeichen gelesen werden, wo nur ein Erneuerungstakt ausgeführt wird.

DMA-Beschreibung	Maschinentakte
Displaylistung	32
Zeichen	4o x 24 = 96o
Zeichendaten	96o x 8 = 768o
Sp.-Erneuerung	262 x 9 - 24x8 = 2166
Gesamtverbrauch:	1o838

Die Summe der DMA pro Gesamtbild beträgt also 1o838 Maschinentakte. Ein Gesamtbild hat 31o Zeilen und insgesamt ca. 29ooo Maschinentakte, so dass für einen Durchlauf im OS-Graphikgang O ca. 36 % der Takte für die DMA verloren gehen.

B. POKEY.

Die Audio-Einrichtung:
Es gibt 4 halb-unabhängige Audiokanäle, jeder mit eigener Steuerung von Frequenz, Geräuschart und Lautstärke. Jeder Kanal besitzt einen 8 Bits grossen "N-Teiler" für die Frequenz, gesteuert von einem 8 - Bit-Register mit der Bezeichnung AUDFX. Weiter hat jeder Kanal ein 8-Bit-Kontrollregister für Geräuschwahl (Polyzähler) und Lautstärke. Dies Register nennt sich AUDCX. (Vgl.Blockdiagramm der Geräusch-Übertragung).

Frequenzteiler.
Alle 4 Frequenzteiler (N-Teiler) können gleichzeitig an die Taktgeber für 64 KHZ oder 15 KHZ angeschlossen werden (AUDCTL - Bit 0). Die Frequenzteiler 1 und 3 lassen sich wahlweise einzeln mit 1,79 MHZ-Takt versorgen (AUDCTL-Bits 6 und 5). Die Teiler 2 und 4 können wahlweise einzeln an den Output der Teiler 1 und 3 gekoppelt werden (AUDCTL-Bits 4 und 3). Dadurch ergeben sich folgende Möglichkeiten: Vier Kanäle mit 8 Bits Auflösung, 2 Kanäle mit 16 Bits Auflösung oder ein Kanal mit 16 Bits und 2 Kanäle mit 8 Bits.

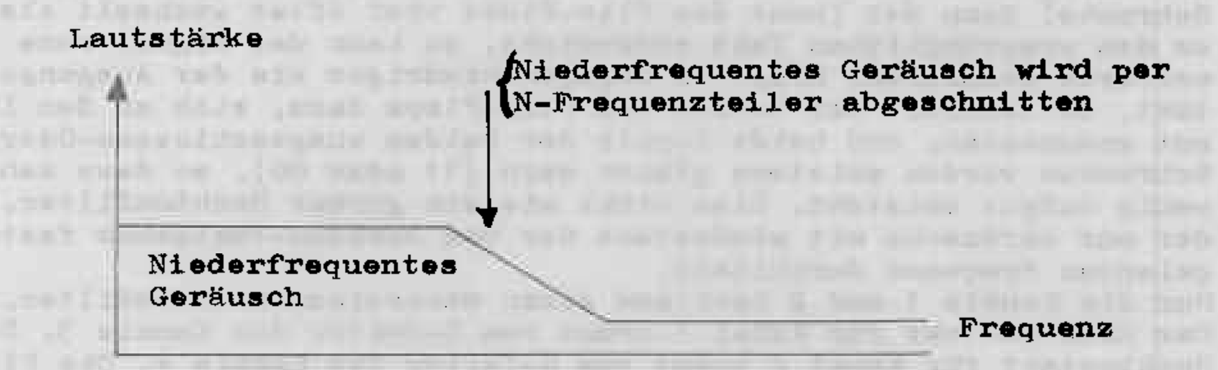
Vielfach-Geräuschzähler.
Es stehen drei variable Zähler (zu 17, 5 und 4 Bits) zur Verfügung, um beliebige Geräusche zu erstellen. Der variable Zähler von 17 Bits kann auf 9 Bits reduziert werden (AUDCTL Bit 7). Diese Zähler sind sämtlich an 1,79 MHZ angeschlossen. Ihre Output-Muster können jedoch individuell mit Hilfe der 4 Audiokanäle nach Massgabe der jeweiligen Frequenzteilers verändert werden. Dadurch entsteht der Eindruck, dass jeder Kanal separate variable Zähler (insgesamt 3 Arten) hat, die speziell auf seine eigene Frequenz eingestellt sind.
Diese Geräuschmuster-Bildung mit variablen Zählern wird über die Bits 5, 6 und 7 jedes AUDCX-Registers gesteuert. Da die Mehrfachzähler (Polycounter) ihr Muster über den N-Frequenzteiler erhalten, kann der Output natürlich nicht schneller variieren als die Musterfrequenz. In diesen (durch Mehrfachzähler geleiteten) Gangarten fungieren die N-Teiler daher gewissermassen als "Niederfrequenz-Filter"* indem sie nur niederfrequente Geräusche durchlassen.

Der Output des Geräusch-Schaltkreises, der oben beschrieben wurde, besteht aus reinen Tönen (quadratische Schwingungsform) oder aus Mehrfachzähler-Geräusch, das eine vom N-Frequenzteiler (Niederfrequenz-Taktgeber) festgelegte Höchstfrequenz besitzt.

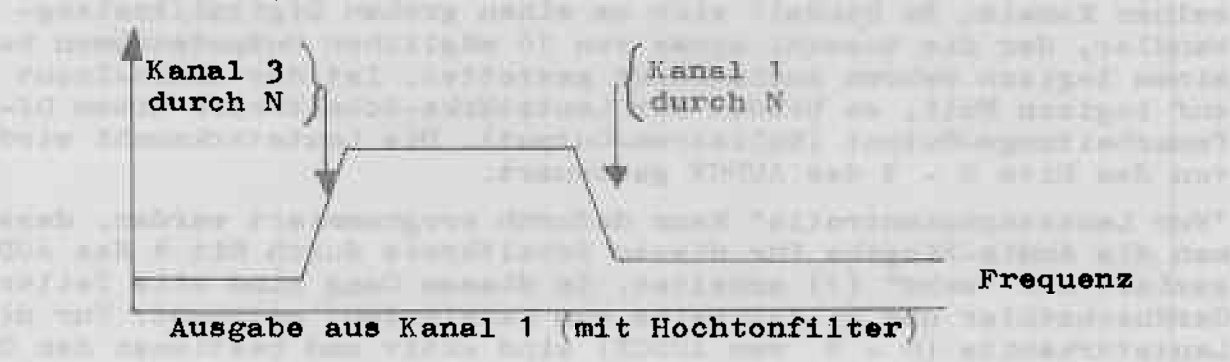
Dieser Output kann auch durch einen Hochtonfilter geleitet werden (AUDCTL-Bits 1 und 2).

* d.h. niedere Frequenzen werden durchgelassen.

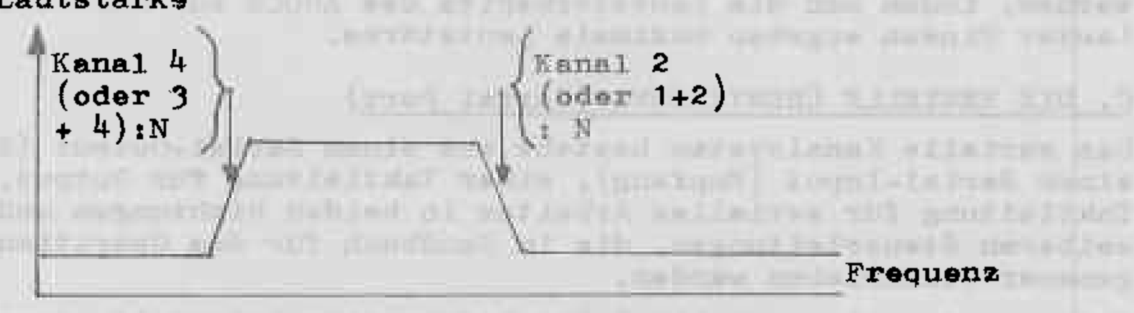
Geräuschfilter



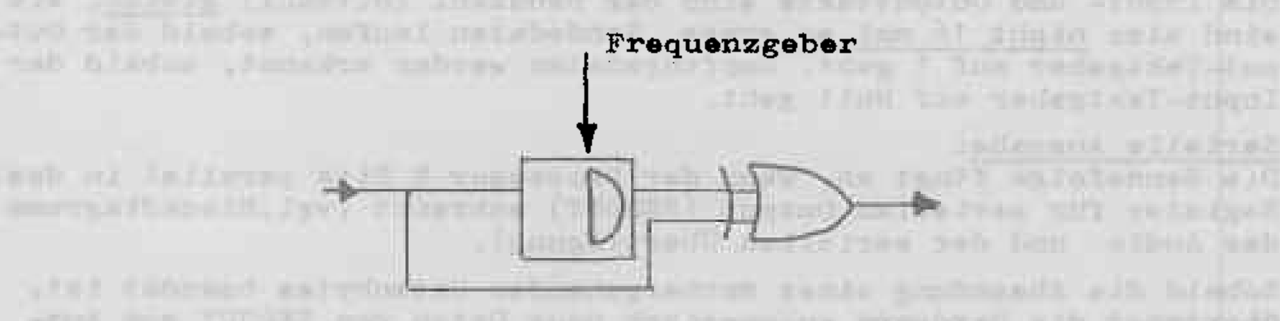
Beliebiges Kanalgeräusch (ohne Hochtonfilter)



Ausgabe aus Kanal 1 (mit Hochtonfilter)



Ausgabe aus Kanal 2 (mit Hochtonfilter)



Hochtonfilter:
Der Hochtonfilter besteht aus dem Flip-Flop "D" und einer "ausgeschlossen-Oder-Schranke"*. Der Output aus dem Geräusch-Schaltkreis wird durch das Flip-Flop strukturiert, und zwar mit einer Frequenz, die von dem "Hochtongeber" bestimmt wird. Input und Output des Flip-Flops laufen durch die erwähnte ausgeschlossen-Oder-Schranke*. Wenn der Input des Flip-Flops viel öfter wechselt als es dem ursprünglichen Takt entspricht, so kann das Signal ohne weiteres passieren. Liegt er dagegen niedriger als der Ausgangstakt, so tendiert der Output des Flip-Flops dazu, sich an den Input anzupassen, und beide Inputs der beiden ausgeschlossen-Oder-Schranken werden meistens gleich sein (11 oder 00), so dass sehr wenig Output entsteht. Dies wirkt wie ein grober Hochtonfilter, der nur Geräusche mit mindestens der vom Hochton-Taktgeber festgelegten Frequenz durchlässt.
Nur die Kanäle 1 und 2 besitzen einen derartigen Hochtonfilter. Der Hochton-Takt für Kanal 1 kommt vom N-Teiler des Kanals 3. Der Hochtontakt für Kanal 2 kommt vom N-Teiler des Kanals 4. Die Filterung ist nur wirksam, wenn Bit 1 oder 2 vom AUDCTL gesetzt ist.

Lautstärke:
Ein Schaltkreis für Lautstärkekontrolle sitzt am Ausgang jedes einzelnen Kanals. Es handelt sich um einen groben Digital/Analaog-Wandler, der die Auswahl eines von 16 möglichen Outputströmen bei einem logisch wahren Audio-Input gestattet. Ist der Audio-Input auf logisch Null, so bildet der Lautstärke-Schaltkreis einen Offenschaltungs-Output (Nullstrom-Output). Die Lautstärkewahl wird von den Bits 0 - 3 des AUDCX gesteuert.

"Nur Lautstärkekontrolle" kann dadurch programmiert werden, dass man die Audio-Eingabe für diesen Schaltkreis durch Bit 4 des AUDCX gezielt auf "wahr" (1) schaltet. In diesem Gang sind alle Teiler, Geräuschzähler und Filterkreise vom Kanaloutput getrennt. Nur die Lautstärkebits (0 - 3 von AUDCX) sind aktiv und bestimmen den Outputstrom des Kanals.

Der Audio-Output jedes einzelnen Kanals kann völlig abgeschaltet werden, indem man die Lautstärkebits des AUDCX auf Null setzt. Lauter Einsen ergeben maximale Lautstärke.

C. DIE SERIELLE ÜBERTRAGUNG (Serial Port)

Das serielle Kanalsystem besteht aus einem Serial-Output (Sendung), einem Serial-Input (Empfang), einer Taktleitung für Output, einer Taktleitung für serielles Arbeiten in beiden Richtungen und aus weiteren Steuerleitungen, die im Handbuch für das Operationssystem genauer beschrieben werden.

Daten werden als serielle 8-Bit-Information behandelt, denen ein Startbit auf logisch Null vorausgeht, während am Ende jeweils ein Stopbit (logisch 1) steht.

Die Input- und Outputtakte sind der Baudzahl (Bitzahl) gleich, sie sind also nicht 16 mal so gross. Sendedaten laufen, sobald der Output-Taktgeber auf 1 geht. Empfangsdaten werden erkannt, sobald der Input-Taktgeber auf Null geht.

Serielle Ausgabe:
Die Sendefolge fängt an, wenn der Prozessor 8 Bits parallel in das Register für seriellen Output (SEROUT) schreibt (vgl. Blockdiagramm des Audio und der seriellen Übertragung).

Sobald die Absendung eines vorhergehenden Datenbytes beendet ist, überträgt die Hardware automatisch neue Daten von SEROUT zum Ausgabe-Verschieberegister, unterbricht den Prozessor, um anzuzeigen

* sogen. Ex-Or-Gatter

dass das SEROUT-Register leer ist (d.h. bereit ist, das nächste Datenbyte zu empfangen) und sendet die Inhalte des Verschiebe - registers in serieller Form und mit angehängten Start- bzw. Stop-Bits. Wenn der Prozessor auf die Unterbrechung reagiert und das SEROUT neu lädt, bevor das Verschieberegister vollständig übertragen ist, so läuft der Sendevorgang weich und gleichmässig ab.

Ausgabedaten werden normalerweise als Logikebenen übermittelt (+4V = wahr, Null V = falsch). Sie können aber auch als Zweiton-Information ablaufen. Dieser Arbeitsgang wird durch Bit 3 des SKCTL eingeschaltet. In ihm wird statt des logischen Wahrheitswertes der Audiokanal 1 gesendet und statt der logischen Null der Audiokanal 2. Kanal 2 muss den niedrigeren Ton enthalten.

Der Prozessor kann die Outputleitung auf Null halten bzw. auf d. Tonkanal 2, wenn der Zweitongang läuft, indem er das Bit 7 des SKCTL setzt. Das ist notwendig, um die Übermittlung des Abbruchcodes (Breakcode) zu ermöglichen (1o Nullen).

Der Takt für serielle Ausgabe:
Die Daten der seriellen Ausgabe wechseln jeweils, sobald der Taktgeber dür serielle Ausgabe auf "wahr" schaltet. Diese Schaltung springt in der Mitte der Ausgabezeit für die jeweilige Bitserie auf Null zurück.

Die Baudrate (Bit pro Zeiteinheit) der Daten und des Taktgebers wird durch Audiokanal 4, Audiokanal 2 oder durch den Inputtaktgeber bestimmt, je nach dem welcher serielle Ablauf durch die Bits 4, 5 und 6 des SKCTL bestimmt wird (vgl.Übersicht am Ende dieses Abschnitts).

Die serielle Eingabe:
Die Empfangssequenz beginnt, sobald die Hardware ein vollständiges serielles 8-Bit-Datenwort plus Start- bzw. Stopbits erhalten hat. Diese Information wird automatisch an das 8 Bit grosse Re - gister für parallele Eingabe geleitet (SERIN), wonach dann der Prozessor anhält, um anzuzeigen, dass ein neues Datenbyte zum Einlesen in SERIN bereitsteht. Der Prozessor muss diese Unterbrechung machen und SERIN bereits lesen, bevor der Empfang des nächsten Eingabeworts abgeschlossen ist, weil sonst ein Fehler "Eingabedaten-Überlauf" gemeldet wird. Dies Ereignis wird ggflls. durch Bit 5 des SSKSTAT signalisiert (wenn Bit 5 des IRQST nicht vor dem Abschluss des nächsten Inputs auf RESET (wahr) zurückgestellt ist). Es bedeutet, dass Eingabedaten verloren gegangen sind. Dies Bit sollte bei jeder Lesung von SERIN getestet werden. Auch das Bit 7 des SKSTAT sollte getestet werden, damit Rahmenfehler ent - deckt werden können, die von zusätzlichen (bzw.fehlenden) Daten - bits stammen.

Direkte serielle Eingabe:
Die Leitung für serielle Dateneingabe kann direkt vom Mikroprozessor gelesen werden, unter Umgehen des Verschieberegisters, indem Bit 4 des SKSTAT abgerufen wird.

Taktgebung für beide Richtungen gleichzeitig:
Diese Taktleitung wird benutzt, entweder um einen Takt von einem aussensender zum Takten von gesendeten oder empfangenen Daten zu verwenden, oder um umgekehrt einen Sende- bzw. Empfangstakt an Außengeräte zu leiten. Die Laufrichtung wird dabei durch den per Bit 4, 5 und 6 des SKCTL gewählten seriellen Gangs bestimmt (vgl. Gangplan am Ende dieses Abschnitts). Sendedaten wechseln jeweils am aufsteigenden Ende dieses Taktes, während Empfangsdaten am abfallenden Ende aufgenommen werden.

Asynchrone serielle Eingabe:
Ungetaktete serielle Daten, deren Dichte in etwa (um 5 % genau) bekannt ist, können in den asynchronen Gängen empfangen werden. Das Verschieberegister für diese Eingabe wird dann durch den Audiokanal 4 getaktet. Die Kanäle 3 und 4 sollten zusammen be - nutzt werden (AUDCTL-Bit 3 = 1), wenn erhöhte Auflösung verlangt wird.
In asynchronen Gängen werden die Kanäle 3 und 4 durch jedes Startbit am Anfang jedes seriellen Datenbytes neu gesetzt. Da - durch kann der serielle Datentakt leicht von dem durch die Ka - näle 3 und 4 bestimmten Takt abweichen.

Die serielle Gangkontrolle:
Es gibt 6 nutzbare (von den möglichen 8) Gänge, die durch die Bits 4, 5 und 6 des SKCTL gesteuert werden. Sie werden weiter unten beschrieben. - Man beachte, dass Zweiton-Output (Bit 3 des SKCTL) in allen diesen Gängen ausser den unteren beiden programmiert werden kann. Die Einschränkung hat ihren Grund darin, dass Kanal 2 zum Festlegen des Übertragungstaktes beim Output ge - braucht wird und deshalb nicht für einen der beiden Töne verfügbar ist.

N.B.: Der Outputtakt entspricht genau der Datengeschwindigkeit beim Output.

Serielle Gangsteuerung (vgl. Beschreibung des SKCTL-Registers):

Erzwungener Abbruch

D7	D6	D5	D4	D3	D2	D1	D0

SKCTL-REGISTER

Paddle-Schnellschaltg. und CTRL-Taste

Zweiton-Steuerung

Gang-Kontrollbits

A= asynchron

D6	D5	D4	Ausg.-Rate	Ausgs. Takt	Eing.-Rate	Doppel-Taktgb.,	Bemerkungen
0	0	0	Ausg	Aus-gg.	Aus-gg.	Ausg./ Eing.	Sende- u.Empf.-Rate wird von aussen festgelegt.Gleichztg. geht interne Taktphase auf 0
0	0	1	Aus-gg.	Aus-gg.	Kanal 4 A	Ausg./ Eing.	Übertraggs.-Rate w.durch externen Taktgeber festgelegt.Empfg. asynchron (Kan.4; Kan.3 + 4)
0	1	0	Kan. 4	Kan. 4	Kan. 4	Kan. 4 Ausgg.	Sende-u.Empf.raten durch Kan.4 festgelegt.Kanal 4 sendet auf Doppel-Taktgeber (2 Richtungen)
0	1	1	Kan. 4 A	Kan. 4 A	Kan. 4 A	Eing.	nicht nutzbar
1	0	0	Kan. 4	Kan. 4	Ausg.	Ausgg. Eingg.	Senderate d.Kan.4 festgelegt, Empfangsrate d.äusseren Taktgeber
1	0	1	Kan. 4 A	Kan. 4 A	Kan. 4 A	Eing.	nicht nutzbar
1	1	0	Kan. 2	Kan. 2	Kan. 2	Kan. 4 Ausg.	Senderate d.Kan.2 festgelegt, Empfangsrate d.KAN.4. Ausgabe von Kanal 4 auf Doppeltaktltg. (2 Ri.)
1	1	1	Kan. 2	Kan. 2	Kan. 4 A	Eing. un-bentzt.	Übertragungsrate d.Kan.2 festgelegt, Empfang asynchron (Kan.3+4). Zweirichtungstakt nicht benutzt (Dreifachbedingung)

Zweitongang (Bit 3) in diesem Zusammenhang nicht anwendbar.

D. DAS UNTERBRECHUNGSSYSTEM

Es gibt zwei Grundtypen von Unterbrechungen im Mikroprozessor: NMI (nicht maskierbare Unterbrechung) und IRQ (Abruf-Unterbrechung). Es wird empfohlen, ein genaues Verständnis dieser Unterbrechungstypen durch das Studium der entsprechenden Kapitel in den Handbüchern zur Hardware und Programmierung des 65o2-Prozessors zu erarbeiten.

In diesem System werden NMIs für Bildprojektion und Bilderneuerung benutzt, IRQs dienen der seriellen Übertragung, Peripheriegerätesteuerung, Zeitgeber und den Tastatureingaben.

Zu den NMIs:
Obwohl diese im Mikroprozessor "nicht maskierbar" sind, besitzt das System dennoch Einschaltbits (Maskenbits) für die NMI-Funktion (Bits 6 und 7 des NMIEN). Gehen diese Bits auf Null, so werden NM-Unterbrechungen verhindert (maskiert), so dass sie den Rechner nicht anhalten können (vgl. Registerbeschreibung für NMIEN). Folgendes sind die drei Arten von NMIs:

1. D7 = Instruktions-Unterbrechung: Während des Projektionsprozesses ruft jede Displayinstruktion mit Bit 7 = 1 diese Unterbrechung hervor (falls sie eingeschaltet ist), sobald die letzte durch die Instruktion projizierte Videozeile beginnt.

2. D6 = Vertikal-Leer-Unterbrechung: Diese Unterbrechung findet (falls eingeschaltet) am Beginn des Vertikal-Leertaktes statt.

3. D5 = Unterbrechung durch Rückstelltaste: Sie wird durch Druck auf die Taste SYSTEM RESET ausgelöst.

Da alle diese Unterbrechungen im Mikroprozessor einen Sprung auf dieselbe NMI-Adresse bewirken, besitzt das System ausserdem noch Statusbits (Zustandsbits) für den NMI, die vom Rechner abgefragt werden können, damit klar wird, was die Unterbrechung veranlasst hat. Die Bits 5, 6 und 7 des NMIST besitzen diese Funktion (s.d.). Die Statusbits werden durch die entsprechende Unterbrechungsfunktion gesetzt, und zwar auch dann, wenn die Unterbrechung durch NMIEN vom Rechner maskiert wurde. Die Statusbits können gemeinsam in Grundstellung gebracht werden, indem man in die Adresse NMIRES schreibt.
Zwei der Unterbrechungs-Einschaltbits (6 und 7 des NMIEN) werden automatisch durch Einschalten der Anlage abgeschaltet (maskiert), damit keine der automatisch nach Einschalten arbeitenden Hilfsroutinen unterbrochen werden können, bevor sie das System startklar gemacht haben+.
Sie können jederzeit vom Prozessor aktiviert werden durch Einschalten von Bits 6 und 7 des NMIEN. Ausser dem Unterbrecher der RESET-Taste können die Bits vom Rechner auch jederzeit wieder abgeschaltet werden durch Laden von Nullen in die Bits 6 oder 7 des NMIEN.
Die Wirkung der RESET-Taste kann nicht abgeschaltet werden, sodass ein unverstellbarer Ausweg aus möglichen Pannensituationen garantiert ist.
Diese NMI-Funktionen sind voneinander zeitlich getrennt, damit Überlappungen vermieden werden, und werden durch die System-Hardware in Impulse umgewandelt, die die von der Rechnerlogik erfor-

---

+ N.B.: Bit 5 wird nie abgeschaltet. Daher darf RESET nicht im Moment des Anschaltens der Anlage gedrückt sein !

derten NMI-Übergänge liefern.

IRQ - Unterbrechungen:
IRQs sind "maskierbar", und zwar alle zusammen durch ein Bit des Statusregisters im Rechner. Dies Bit wird automatisch auf die Abschaltstellung gebracht, sobald die Maschine eingeschaltet ist (s.oben) ++. Zusätzlich zu diesem IRQ-Maskenbit gibt es separate Einschaltbits für jede IRQ-Funktion (Bits 0 - 7 von IRQEN). Diese Bits werden nicht durch die Einschaltung des Systems aktiviert, müssen also vom Programm aktiviert werden, bevor man den IRQ des Rechners einschaltet. Es gibt folgende 8 Arten von IRQs:

D7 = BREAK (Druck auf die Breaktaste),
D6 = ANDERE TASTEN (Druck auf irgend eine andere Taste),
D5 = SERIELLE EINGABE BEREIT (Byte von seriellen Daten wurde empfangen und steht zur Lesung durch den Rechner im SERIN-Register bereit),
D4 = SERIELLER OUTPUT BENÖTIGT (Byte von seriellen Daten wird übertragen, und SEROUT steht für neue Auffüllung durch den Rechner bereit,
D3 = ÜBERTRAGUNG BEENDET (die serielle Datenübermittlung ist beendet, das Outputregister ist leer),
D2 = TIMER NR.4 (Audio-Teiler 4 hat bis Null zurückgezählt),
D1 = TIMER NR.2 (Audio-Teiler 2 hat bis Null zurückgezählt),
D0 = TIMER NR.1 (Audio-Teiler 1 hat bis Null zurückgezählt),

Zusätzlich zu den erwähnten IRQs (eingeschaltet durch Bits 0-7 des IRQEN und erkannt durch die Statusbits 0-7 des IRQST gibt es noch zwei Syste -IRQs, die über den Antrieb bzw. den Unterbrecher des seriellen Übertragungskanals erzeugt werden:

D7 des PACTL = Statusbit für periphere Unterbrechung "A",
D0 des PACTL = Schaltbit für periphere Unterbrechung "A",
D7 des PBCTL = Statusbit für periphere Unterbrechung "B",
D0 des PBCTL = Schaltbit für periphere Unterbrechung "B".

Die letzteren beiden Unterbrechungen werden automatisch ausgeschaltet, wenn man den Hauptschalter einschaltet, und ihre Statusbits werden durch Lesen aus dem Kanal-A-Register und dem Kanal-B-Register neu gesetzt (vgl. PORT A, PACTL, PORT B und PBCTL).

Das IRQEN-Register schaltet, wie das NMIEN-Register, Unterbrechungen ein, wenn seine Bits auf 1 stehen (auf logisch wahr). Das IRQST dagegen, anders als das IRQST, besitzt Unterbrechungs-Statusbits, die normalerweise auf logisch wahr stehen und nur auf Null gehen, wenn eine Unterbrechung angefordert wird ("Abruf"). Die Statusbits des IRQST werden erst auf logisch wahr zurückgestellt, wenn eine Null in das entsprechende IRQEN-Bit geschrieben worden ist. Das schaltet dann die Unterbrechung ab und setzt gleichzeitig das Statusbit für die Unterbrechung auf 1.

Bit 3 des IRQST ist keine Sperre und wird nicht durch Unterbrechungsabschaltung neu gesetzt. Es steht auf Null, wenn der serielle Ausgang leer ist (serielle Ausgabe beendet) und auf 1, falls nicht.

++ N.B.: Ein NMI schaltet dies Bit ebenfalls ab.

Unterbrechungstechnik
- Zusammenfassung -

Name	Funktionen	Einschaltung	Status	Status-Rückschaltung
NMI-Unterbrechungen	Display-Instruktion Vert.-Leertakt, Rückstellkn.	NMIEN (Bits 6-7) Normalerweise auf Null (abgeschalt.)	NMIST (Bits 5-7) norm.weise 0 (keine Unbrechg.)	Adress-NMIRES (stellt alle NMI-Statusbits zurück)
IRQ-Unterbrechungen	TASTEN Serielle Kanäle Zeitgeber	IRQEN (Bits o-7) (Null bedeutet abgeschaltet) *	IRQST (Bits o - 7) norm.weise "wahr" (keine Unterbr.)	Rückstellung (auf "wahr") durch 0 in d. entspr. Bit v. IRQEN (ausser Bit 3) *
	Peripherie-Kanal A	D 0 von PACTL normalerw. 0 (abgeschalt.)	D 7 v.PACTL norm.weise 0 (keine Unterbrchg.)	Rückstellg.d. Lesen d.Reg.s Port A
	Peripherie-Kanal B	D 0 v.PBCTL normalerw.0 (abgeschalt.)	D 7 v.PBCTL norm.weise 0 (keine Unterbrchg.)	Rückstellg.d. Lesen d. Registers Port B

E. STEUERGERÄTE

Mit den den 4 Steckdosen an der Vorderseite der Konsole kann eine Anzahl von Steuergeräten verbunden werden, z.B. Spielhebel, Paddles (Drehpotentiometer, "Pots"), 12er-Tastaturen und Lichtstifte (falls vorhanden).

Die Steuerkanäle werden über die Register PORT A, PORT B so wie POT und TRIG gelesen. Das OS liest sie während des Vertikal-Leertaktes und speichert die Informationen in seinen eigenen RAM-A - dressen. Diese lauten: STICK, PADDLO bis PADDL7, PTRIG und STRIG. Das OS bereitet PORT A und PORT B für die Eingabe vor und zwar dadurch, dass Bit 2 von PACTL oder PORT B (Kanalsteuerung) auf Null gesetzt wird (Auswahl des Registers für die Richtungskontrolle) und dass dann der gewünschte Kanal mit Nullen versehen wird.

Bit zwei von PACTL (PBCTL) wird anschliessend auf eine 1 zurückgeschaltet, wodurch das Programm jetzt aus dem Kanal lesen kann. Die Steuerkanäle können auch auf Ausgabe eingestellt werden, indem man Einsen statt Nullen schreibt, während der Gang für die Richtungskontrolle gewählt wird.

<u>Spielhebel (Joysticks):</u>
Sie haben 4 Schalter, je einen für rechts (R), links (L), rückwärts (B) und vorwärts (F). Diese Schalter werden über PORT A und PORT B gelesen. Ein fünfter Schalter wird wirksam, wenn man den roten Knopf drückt. Dieser Auslöser wird von den Registern TRIGO bis TRIG3 gekesen. 0 zeigt dabei an, dass der entsprechende Knopf gedrückt wurde, 1, dass dies nicht der Fall ist.

Die TRIG-Register werden normalerweise direkt gelesen, sie können jedoch auch mit Sperre benutzt werden. Wenn man Bit 2 von GRACTL auf Null bringt, wird die Sperre gelöst und geht auf 1.

Der Eintrag einer Eins in das genannte Bit 2 lässt die Sperre wirksam werden. Wenn der Auslöser eines Spielstabes gedrückt wird, während Bit 2 von GRACTL auf 1 steht, wechselt der Sperrwert auf Null und bleibt dort. Im Programm nützt man dies aus, um zu bestimmen, ob die Auslöser innerhalb einer bestimmten Zeitspanne überhaupt benutzt worden sind.

Die Paddles:
Paddles werden paarweise verwendet, so dass man 8 Paddles in den 4 Steckdosen unterbringen kann. Ihre Werte werden zunächst in POTGO gespeichert, während die POT-Register frühestens 228 Zeilen später gelesen werden. Die Werte reichen von 0 (Paddle nach rechts gedreht) bis 228 (Paddle nach links gedreht). Der Wert zwigt an, nach wievielen Bildzeilen der an das Pontentiometer angeschlossene Messkondensator (Capacitor) aufgeladen ist. Bewegen des Drehknopfes nach rechts vermindert den Widerstand, so dass schnell geladen wird; Knopfdrehung nach links erhöht den Widerstand und damit die Lade - zeit. Die Entladetransistoren werden zum Entladen der Kondensatoren eingesetzt, wenn eine neue Ablesung beginnen soll.

Der Befehl POTGO löscht die Zähler und schaltet die Entladetransistoren ab, damit die Kondensatoren sich neu laden können. Des Register ALLPOT enthält ein Bit pro Paddle. Hat der Messkondensator sich bis zum Schwellenwert aufgeladen, so wechselt das ALLPOT-Bit von 1 auf 0 und das Register POT enthält die korrekten Werte.

Bit 2 von SKCTL (serielle Übertragungs-Steuerung) schaltet die schnelle Paddleverbindung ein. In diesem Gang werden nur zwei Bildzeilen benötigt, um die Messkondensatoren auf Höchstwert zu bringen.
Zuerst wird hier Bit 2 auf Null gesetzt, um die Kondensatoren zu leeren. Dann wird Bit 2 wieder auf 1 gebracht, um die POT-Übertragung zu starten. Die schnelle Paddlepot-Übertragung ist nicht so genau wie der normale Übertragungsgang für die Paddles.

Bit 2 des SKCTL muss bei normalem Paddlegang auf Null gesetzt werden, da sonst die Kondensatoren nicht geleert würden. Man beachte, dass einige Paddles einen Bereich von unter 228 haben auf Grund v. Unterschieden in den Pots. Die linken und rechten Paddle-Auslöser jedes Paddlepaares werden von den linken und rechten Bits für die entsprechenden Spielstäbe gelesen (PORT A oder PORT B).

Tastatursteuerung:
Jede Einsteck- (Zusatz-) Tastatur hat ein Feld mit 12 Tasten und wird an eine der Steckdosen für die Spielstäbe angeschlossen. Der erste Schritt beim Benutzen der Tastatur besteht darin, eine Reihe zu wählen (vgl. PORT A, nächster Abschnitt). In die anderen Reihen sollten Einsen geschrieben werden. Spalten werden über die Register POT und TRIG gelesen (vgl.Kontakttabelle im nächsten Abschnitt).

Der Anhang H des BASIC-Handbuches enthält ein BASIC-Programm, das die Tastatursteuerung liest. Die erste und die zweite Spalte der Tastatur benutzen dieselben Kontaktausgänge wie die Pots der Paddlesteuerung, sie werden also über die POT- (bzw.PADDL-) Register empfangen.

Sobald eine Taste gedrückt wird, ist die Potleitung kurzgeschlossen, die Potkondensatoren laden deshalb nie bis zum Schwellenwert auf und die Lesung beträgt 228 (das Maximum).

Wenn der Knopf in der gewählten Reihe bzw. Spalte nicht gedrückt wird, schaltet der Kondensator über einen relativ kleinen Widerstand auf +5 Volt und gibt einen Potwert von 2 an (dies kann variieren).
Da die Ablesung nicht kritisch ist, kann der schnelle Potgang benutzt werden, so dass eine Wartezeit von nur zwei Zeilen zwischen der Wahl der Reihen und der Lesung des Potregisters erforderlich ist.
Man hat sich angewöhnt, die Pot-Ablesung mit 10 zu vergleichen (dezimal). Ist sie grösser als 1o, dann muss der Knopf gedrückt worden sein. Die dritte Spalte wird über die Leitung für den Auslöser des Spielstabes gelesen, so dass sie genau wie ein solcher Auslöser wirkt (Null = Knopf gedrückt, Eins = nicht gedrückt).

<u>Der Lichtstift:</u>
Das ist ein Gerät, das den Bildstrahl bei seinem Ablauf über den Monitor registrieren kann. Es wird benutzt, um direkt auf ein Objekt auf dem Bildschirm zu zeigen. Z.B. kann man mit dem Lichtstift Titel aus einem Menu heraussuchen oder Linien ziehen.

Die ATARI 4oo/800 ist so gebaut, dass ein Lichtstift an eine der Steuersteckdosen angeschlossen werden kann (vgl.Ende des nächsten Abschnitts).
Wenn irgend eine der Leitungen der Spielhebel-Auslöser (Ausgang 6) angesprochen wird, nimmt der ANTIC-Chip den laufenden Wert von VCOUNT und speichert ihn in PENV. Der Wert des horizontalen Bild - taktes (Color Clock)(0 - 227 dezimal) wird in PENH gespeichert. Das untere Bit ist unpassend und sollte ignoriert werden.

Da das Display und die Änderung des Lichtstiftregisters eine Anzahl von Verzögerungen beinhalten, muss jedes TV-System vor Lichtstifteinsatz justiert werden. Software, die den Lichtstift benutzt, sollte eine vom Benutzer prüfbare Justageroutine enthalten. So könnte der Benutzer z.B. mit dem Stift auf ein Fadenkreuz in der Bildmitte zeigen, während das Programm den horizontalen Abstand errechnet. PENH macht dann eine Bildverschiebung von 227 bis 0, nahe an die rechte Kante des Bildrandes (wegen der Verzögerung). Der Stift arbeitet nicht, wenn er auf ein schwarzes Bildfeld gerät, weil da der Bildstrahl ausfällt. Es ist ratsam, zwei Messwerte zu lesen und davon den Durchschnitt zu nehmen, da der Benutzer den Stift wohl nicht absolut ruhig halten kann.

II. Die Hardware-Register

Dieser Abschnitt gibt eine Liste der Hardware-Register und der Schattenregister des Operationssystems (OS) wieder.

Im folgenden bedeutet "wahr" ein auf "1" stehendes Bit.

A. PAL (D014)

nicht benutzt	D3	D2	D1	nicht ben.

D3	D2	D1	
1	1	1	NTSC (U.S.A.-Fernsehen)
0	0	0	PAL (TV-System in der BRD)

Dies Byte kann durch ein Programm darauf geprüft werden, nach welchem TV-System gearbeitet wird.

B. Unterbrechungskontrolle.

NMIEN (Nicht maskierbare Unterbrechung wird eingeschaltet)(D40E):

Diese Adresse schreibt Daten in die Einschaltbits für die NM-Unterbrechung.

0 = ausgeschaltet (maskiert)

1 = eingeschaltet

D7	D6	nicht benutzt

D7 Schaltet die Unterbrechung der Displaylisten-Instruktion ein. Dies Bit wird durch Neuanschalten der Maschine gelöscht und kann im übrigen durch den Prozessor gesetzt oder gelöscht werden.

D6 Schaltet die Vertikal-Leertakt-Unterbrechung ein. Dies Bit wird durch Neuanschalten der Maschine gelöscht und kann im übrigen durch den Prozessor gesetzt oder gelöscht werden.

SYSTEM RESET - Unterbrechung:

Diese Unterbrechungsmöglichkeit ist immer wirksam. Die SYSTEM RESET-Taste darf beim Einschalten der Maschine nicht gedrückt sein !

(wird durch den IRQ-Code des Operationssystems auf hexadez. 40 gesetzt)

NMIST (Status der nicht maskierbaren Unterbrechung)(D40F):

Diese Adresse liest das NMI-Statusregister ab (Lesung durch den NMI-Code des Oper.Systems).

0 = keine Unterbrechung
1 = Unterbrechung

D7	D6	D5	unbenutzt

D7 identifiziert eine NMI, hervorgerufen durch Bit 7 einer Displaylisten-Instruktion.

D6 identifiziert eine NMI, hervorgerufen durch den Beginn eines Vertikal-Leertaktes.

D5 identifiziert eine NMI, hervorgerufen durch Druck auf die SYTSEM RESET - Taste.

NMIRES (Neueinstellung des Statusregisters)(D40F):

Diese Adresse ist nurschreibend. Sie stellt das Statusregister für die nicht maskierbare Unterbrechung (NMIST) neu ein.

nicht benutzt

(Einschreibung erfolgt durch den NMI-Code des Oper.Systems).

IRQST (IRQ-Unterbrechungsstatus)(D20E): Diese Adresse liest die Daten aus dem Statusregister für die IRQ-Unterbrechung.

0 = Unterbrechung,
1 = keine Unterbrechung

D7	D6	D5	D4	D3	D2	D1	D0

D7 = 0 : Unterbrechung durch Druck auf BREAK-Taste,
D6 = 0 : Unterbrechung durch irgend eine andere Taste,
D5 = 0 : Unterbrechung, weil Daten für serielle Eingabe bereit,
D4 = 0 : Unterbrechung, weil Daten für serielle Ausg. angefordert.
D3 = 0 : Unterbrechung, weil serielle Byteübertrgg. nach aussen beendet.*
D2 = 0 : Unterbrechung durch Zeitgeber 4.
D1 = 0 : Unterbrechung durch Zeitgeber 2.
D0 = 0 : Unterbrechung durch Zeitgeber 1.

---

* wird für die Erzeugung zweier Stopbits benutzt; vgl.Beschreibung des IRQ im vorigen Kapitel (keine direkte Neueinstellung auf Bit 3)

IRQEN (Einschaltung der IRQ-Unterbrechung)(D20E):

Diese Adresse schreibt Daten in die Bits, die die IRQ-Unterbrechung einschalten.

0 = Ausschaltung (das entspr.IRQST-Bit wird 1),
1 = Einschaltung.

D7	D6	D5	D4	D3	D2	D1	D0

D7 Einschaltung der Unterbrechung durch BREAK-Taste,
D6 " " " " andere Tasten,
D5 " " " wegen bereitstehender Eingabe,
D4 " " " wegen Anforderung einer Ausgabe,
D3 " " " wegen Beenden d. seriellen Ausg.,
D2 " " " durch Timer Nr.4,
D1 " " " durch Timer Nr.2,
D0 " " " durch Timer Nr.1.

Die OS-Schattenadresse POKMSK (hexadez.10).
Es sind UND und ODER zuprogrammieren, um ein Bit im POKMSK ohne Berührung der übrigen Bits zu ändern. Der gewünschte Wert muss sowohl im IRQEN wie im POKMSK gespeichert werden.

C. BILDZEILEN-KONTROLLE.

VCOUNT (Vertikalzähler)(D40B): Diese Adresse liest den Bildzeilenzähler (dessen 8 höchste Bits).

D7	D6	D5	D4	D3	D2	D1	D0	
V8	V7	V6	V5	V4	V3	V2	V1	V0

V0 wird nicht gelesen. Die Auflösung ist zweizeilig.

WSYNC (Warte auf Synchronisation bei horizontalem Leertakt, d.h. warte, bis die nächste TV-Zeile beginnt (D40A):

nicht benutzt

Diese Adresse legt eine Sperre ein, welche die RDY-Leitung zum Mikroprozessor auf Aus hält, so dass der Rechner wartet, bis die Sperre durch den Beginn des Horizontal-Leertaktes aufgehoben wird. Display-Unterbrechungen können durch den Gebrauch des WSYNC um eine Zeile verzögert werden (wird auch von der "Klick"-Routine der Tastatur benutzt).

D. DIE GRAPHIK-KONTROLLE.

DMACTL (Kontrolle der direkten Speicheradressierung)(D400):
Diese Adresse schreibt in das Kontrollregister für DMA ein.

nicht benutzt	D5	D4	D3	D2	D1	D0

D5 = 1 : Einschaltung der DMA für Instruktionsabruf,
D4 = 1 : einzeilige Auflösung für Spieler/Geschosse,
D4 = 0 : zweizeilige Auflösung " " " ,
D3 = 1 : Einschaltung der DMA für Spieler,
D2 = 1 : " " " " Geschosse,
D1,D0= 0 0 : Keine DMA für Spielfeld,
0 1 : DMA für enges Spielfeld (128 Clocks),
1 1 : DMA für breites Spielfeld (192 Clocks).

Vgl. GRACTL. OS-Schattenreg.: SDMCTL (22F), Auslassungswert (default value) ist hexadezimal 22.

GRACTL (Graphikkontrolle)(D01D): Diese Adresse schreibt Daten in die Graphikkontrollregister.

nicht benutzt	D2	D1	D0

D2 = 1 : Einschaltsperren auf TRIGO - TRIG3 - Eingaben (die Sperren werden aufgehoben und TRIGO - TRIG3 arbeiten normal, wenn dies Kontrollbit auf 0 steht).

D1 = 1 : Einschaltung der DMA für Spieler zur Übertragung an die Graphikregister für Spieler,

D0 = 1 : Einschaltung der DMA für Geschosse zur Übertragung an die Graphikregister für Geschosse.

Die DMA wird durch Setzen von Bits sowohl im DMACTL wie im GRACTL eingeschaltet. Das Aktivieren von DMACTL allein würde zwar den Verlust von Maschinentakten aber keinerlei Projektion zur Folge haben.

CHACTL (Zeichenkontrolle)(D401): Diese Adresse schreibt Daten in die Kontrollregister für Schriftzeichen.

nicht benutzt	D2	D1	D0

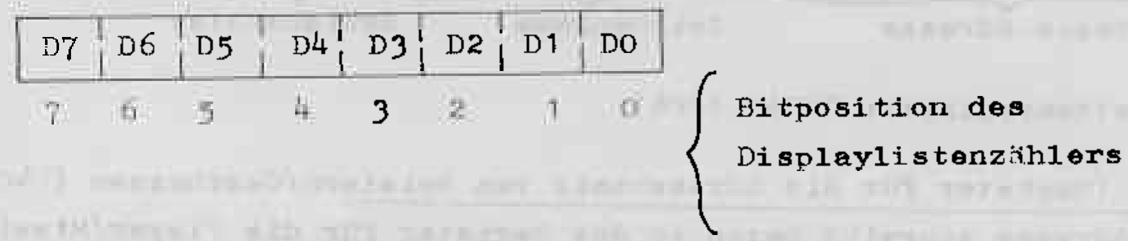
D2 : Bit für vertikale Reflexion von Zeichen.- Dies Bit wird am Anfang jeder Schriftzeile abgefragt. Wenn gesetzt, veranlasst es, dass die entsprechende Schriftzeile vertikal reflektiert wird bzw. dass die Zeichen auf dem Kopf stehen.

D1 : Signal für Video-Umkehrung bei Zeichen (nur beim 4o-Zeichengang). Wenn Bit 7 des Zeichencodes wahr ist, so veranlasst dies Signal, dass die Zeichen blau auf weiss erscheinen (wenn die normalen Farben weiss auf blau sind).

D0 : Signal für Zeichen-Ausblendung (nur beim 4o-Zeichen-Gang). Wenn Bit 7 des Zeichencodes wahr ist, so veranlasst dies Signal die Ausblendung des betreffenden Zeichens. Wenn man Bit 7 des Zeichens auf 1 setzt und D0 im CHACTL periodisch umschaltet, so blinkt das Zeichen periodisch auf.

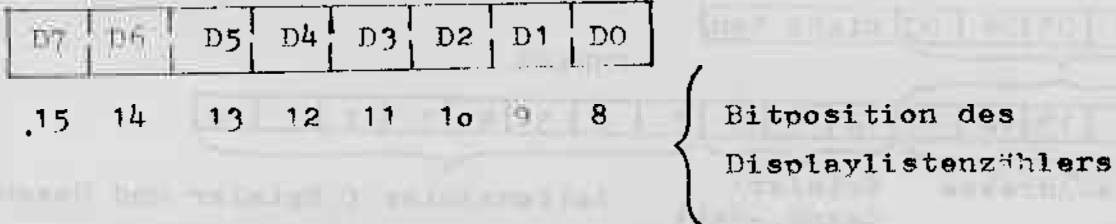
OS-Schattenregister: CHACT (2F3).

DLISTL (Displaylisten-Unterteil (D402): Diese Adresse schreibt Daten in das untere Byte des Displaylistenzählers.



OS-Schattenregister: SDLSTL (hexadez.230)

DLISTH (Displaylisten-Oberteil)(D403): Diese Adresse schreibt Daten in das obere Byte des Displaylisten-Zählers.



OS-Schattenregister: SDLSTH (hexadez.231).

Die Displayliste ist eine Liste von Projektions-Instruktionen im Speicher. Diese werden vom Displaylistenzähler nacheinander abgerufen. Durch Laden der Instruktionsregister definiert man auch den Anfang der Displayliste (vgl.vorige Abschnitte).

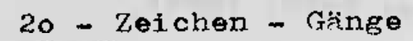
Anmerkg.: Die jeweils oberen 6 Bits dienen nur als Sperren und haben keine Zähleigenschaft. Deshalb kann die Displayliste nie eine 1-K-Byte-Grenze überschreiten, ohne dass eine Sprunginstruktion angewandt wird!

DLISTL und DLISTH sollten nur während des Vertikal-Leertaktes bzw. bei abgeschalteter DMA geändert werden. Andernfalls kann das Bild anfangen zu rollen. Bit 7 des NMIEN muss gesetzt sein, um Displaylisten-Unterbrechungen empfangen zu können.

CHBASE (Register für die Adressbasis von Zeichen)(D409):

Diese Adresse schreibt Daten in das Basisregister der Zeichenadresse. Die Information spezifiziert das höchste Byte (MSB) der Adresse des gewünschten Zeichensatzes (vgl.voriges Kapitel). Man beachte, dass die letzten 1 oder 2 Bits auf Null stehen sollen.





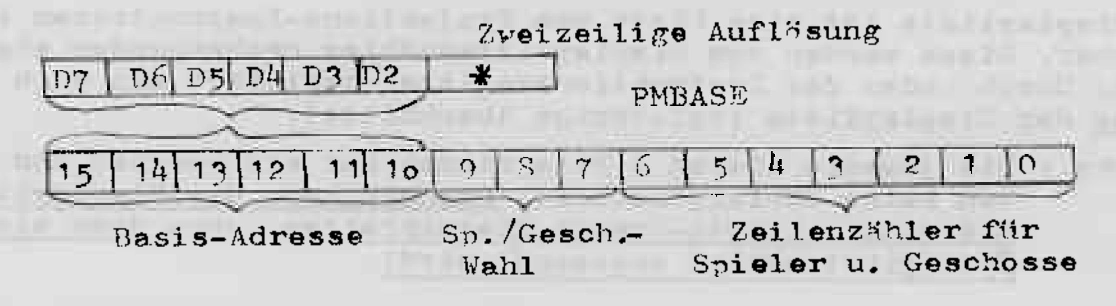


OS-Schattenregister: CHBAS (2F4)

PMBASE (Register für die Adressbasis von Spielern/Geschossen (D407)

Diese Adresse schreibt Daten in das Register für die Player/Missile-Adressbasis. Die Daten geben an, wo das MSB (höchstes Byte) der Adresse mit den Daten für die Spieler- und Geschoss-DMA steht (vgl. voriges Kapitel).





---

* nicht benutzt

HSCROL (Register für horizontale Verschiebung)(D404): Diese Adresse schreibt Daten in das Register f. Horizontalverschiebung. Nur das Spielfeld wird verschoben, nicht Spieler und Geschosse !

nicht benutzt	D3	D2	D1	D0

Rechtsverschiebung um 0-15 Color-Clocks.

Das Display wird um soviele Color- Clocks nach rechts verschoben, wie im HSCROL für jede Displaylisten-Instruktion mit einer 1 in ihrem HSCROL-Signalbit angegeben (Bit 4 des Instruktionsbytes).

Wenn horizontale Verschiebung eingeschaltet ist, werden mehr Datenbytes gebraucht. Für enges Spielfeld (vgl.DMACTL, Bits 1 und 0) sollte dieselbe Anzahl von Bytes pro Zeile zur Verfügung stehen wie für normales Spielfeld ohne Verschiebung.

Ähnlich sollte man für ein normales Spielfeld dieselbe Anzahl von Bytes benutzen wie für ein breites ohne Verschiebung. Für ein breites Spielfeld kommt kein Wechsel in der Byteanzahl in Frage und es wird Hintergrundfarbe eingeschoben.

VSCROL (Register für vertikale Verschiebung)(D405): Diese Adresse schreibt Daten in das Register für vertikale Verschiebung.

nicht benutzt	D2	D1	D0

Gänge mit 8 Bildzeilen.

nicht benutzt	D3	D2	D1	D0

Gänge mit 16 Bildzeilen.

Die Projektion wird aufwärts verschoben, und zwar um soviele Zeilen, wie das VSCROL für jede Displaylisten-Instruktion mit einer 1 im Signalbit für das VSCROL angibt (Bit 5 des jeweiligen Instruktionsbytes). Der verschobene Bereich endet bei der ersten Instruktion, die eine 0 in Bit 5 trägt (vgl. vorigen Abschnitt).

PRIOR (Priorität)(D01B): Diese Adresse schreibt Daten in das Register für die Prioritätskontrolle.

D7	D6	D5	D4	D3	D2	D1	D0

D7 - D6 = 0, D5 = Einschalten des mehrfarbigen Spielers.
Dies Bit setzt die Funktion logisches Oder der Farbbits für Spieler 0 zusammen mit Spieler 1, desgleichen für Spieler 2 zusammen mit Spieler 3.- Das ermöglicht ein Überlappen von 2 Spielerpositionen mit einer Auswahl aus drei Farben für die jeweils überlappte Zone. -

D4 : Einschalten des fünften Spielers:
Dieses Bit veranlasst alle Geschosse, die Farbe der Spielfeldart 3 anzunehmen (COLPF3). Dadurch können die Geschosse eine gemeinsame Farbe erhalten und zu einem fünften Spieler zusammengefasst werden.

D3, D2, D1 u. D0: Prioritätswahl (gegenseitig ausgeschlossen).
Diese Bits wählen eine von 4 Prioritätsarten. Objekte mit höherer Priorität erscheinen jeweils vor solchen mit niedrigerer Priorität, diese überdeckend.

Höhere Priorität ↑

D3=1	D2=1	D1=1	D0=1
PF0	PF0	P0	P0
PF1	PF1	P1	P1
P0	PF2	PF0	P2
P1	PF3 + P5	PF1	P3
P2	P0	PF2	PF0
P3	P1	PF3 + 5	PF1
PF2	P2	P2	PF2
PF3 + P5	P3	P3	PF3 + 5
BAK	BAK	BAK	BAK

Anmerkung: Der Gebrauch von Prioritätsbits in einer nicht-ausschließenden Weise (mehr als 1 Bit aus "wahr") führt zu Objekten, die in einem Prioritätskonflikt stehen und die dann in der Überlappungszone schwarz erscheinen.

Beispiel: PRIOR-Code = 1010: dies lässt P0 oder P1 schwarz erscheinen, wenn sie über PF0 oder PF1 geraten. Ebenso werden P2 oder P3 schwarz, wenn sie über PF2 oder PF3 rutschen.
In den Gängen mit einer Farbe und 4o Zeichen wird die Leuchtstärke eines Pixels in einem Zeichen durch COLPF1 bestimmt, ohne Rücksicht auf die Priorität. Wenn ein Spieler oder ein Geschoss mit höherer Priorität das Zeichen überdeckt, dann wird die Farbe dieser Zone von der Spielerfarbe bestimmt.
OS-Schattenregister: GPRIOR (26F).
COLPF0 - COLPF3 (Spielfeldfarbe)(D016, D017, D018, D019): Diese Adressen schreiben Daten in die Register für Farbe und Lichtstärke des Spielfeldes.

D7	D6	D5	D4	D3	D2	D1	D0

(vgl. COLBK wegen der Bitzuordnung)

OS-Schattenregister: COLOR0 - COLOR3 (2C4 - 2C7).

COLBK (hintergrundfarbe)(D01A): Diese Adresse schreibt Daten in das Register für Farbe und Helligkeit des Hintergrundes.

Farbe				Helligkeit				
D7	D6	D5	D4	D3	D2	D1	unbenutzt	
X	X	X	X	0	0	0		Helligkeit 0 (schwarz)
				0	0	1		
					ETC.			
				1	1	1		maximale Helligk. (weiss)
0	0	0	0	Grau				
0	0	0	1	Gold				
0	0	1	0	Orange				
0	0	1	1	Rot-Orange				
0	1	0	0	Rosa				
0	1	0	1	Purpur				
0	1	1	0	Violett				
0	1	1	1	Blau				
1	0	0	0	Blau				
1	0	0	1	Hellblau				
1	0	1	0	Türkis				
1	0	1	1	Blaugrün				
1	1	0	0	Grün				
1	1	0	1	Gelbgrün				
1	1	1	0	Grün-Orange				
1	1	1	1	Hellorange				

OS-Schattenregister: COLOR4 (2C8)

E. SPIELER UND GESCHOSSE

DMACTL, GRACTL, PMBASE und PRIOR berühren auch Spieler und Geschosse.

COLPM0 - COLPM3 (Farbe von Spielern und Geschossen)(D012 - D015): Diese Adressen schreiben in die Register für Farbe und Helligkeit von Spielern und Geschossen. Geschosse besitzen dieselbe Farbe und Helligkeit wie ihr zugehöriger Spieler, abgesehen von dem aus den Geschossen gebildeten 5. Spieler.- Der ggflls. aus Gesch. gebildete 5. Spieler bekommt seine Farbe vom COLPF3.

D7	D6	D5	D4	D3	D2	D1	D0

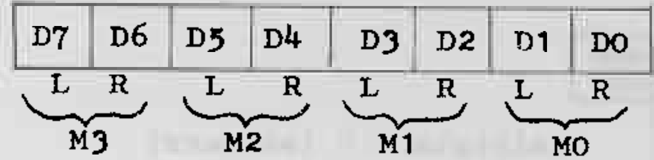
(vgl. COLBK wegen d. Bitzuordnung)
OS-Schattenregister: PCOLR0 - PCOLR03 (2C0 - 2C3).

GRAFP0 - GRAFP3 (Spieler-Graphikregister)(P0 D00D, P1 D00E, P2 D00F, P3 D010): Diese Adressen schreiben Daten direkt in die Graphikregister für Spieler, unabhängig von der DMA. Wenn die DMA angeschaltet ist, werden die Graphikregister automatisch aus dem vom PMBASE angegebenen Speicherbereich geladen (vgl. voriges Kapitel).

D7	D6	D5	D4	D3	D2	D1	D0
Linker							Rechter

Spieler auf d.Bild

GRAFPM (Graphikregister für Geschosse)(D011): Diese Adresse schreibt Daten direkt in das Graphikregister für Geschosse, unabhängig von der DMA.



SIZEP0 - SIZEP3 (Spielergrösse)(P0 D008, P1 D009, P2 D00A, P3 D00B): Diese Adressen schreiben Daten in die Kontrollregister für die Spielergrösse.

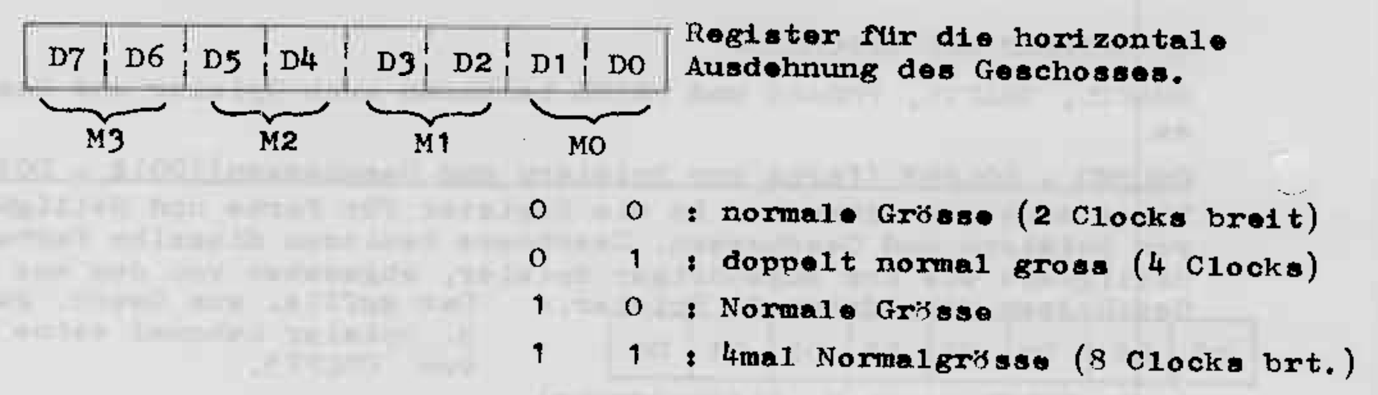
nicht benutzt	D1	D0

Reg. f. horiz. Ausdehng. (Spieler)

D1	D0	
0	0	: normale Grösse (8 Clocks breit)
0	1	: das Doppelte d.norm. Grösse (16 Clocks breit)
1	0	: normale Grösse
1	1	: 4mal Normalgrösse (32 Clocks)

Bei normal grossen Objekten entspricht jedes Bit im Graphikregister einem Color Clock. Für breitere Objekte wird jedes Bit über mehr als ein Color Clock ausgedehnt.

SIZEM (Geschossgrösse)(D00C): Diese Adresse schreibt Daten in das Kontrollregister für die Geschossgrösse.



0	0	: normale Grösse (2 Clocks breit)
0	1	: doppelt normal gross (4 Clocks)
1	0	: Normale Grösse
1	1	: 4mal Normalgrösse (8 Clocks brt.)

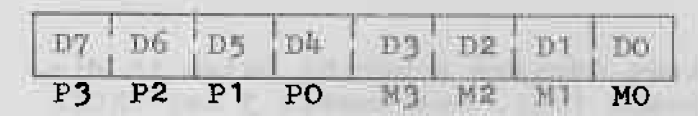
HPOSP0 - HPOSP3 (Horizontal-Position des Spielers)(P0 D000, P1 D001, P2 D002, P3 D003): Diese Adressen schreiben Daten in das Register für die Horizontalposition des Spielers. Der Wert bestimmt die Colorclock-Stellung der linken Objektkante. Hexadez.3o ist die Lage der linken Bildkante beim normalen Monitor, hexadez.0 die Lage von dessen rechter Bildkante.

D7	D6	D5	D4	D3	D2	D1	D0

HPOSMO - HPOSM3 (Horizontalposition der Geschosse)(MO D004, M1 D005, M2 D006, M3 D007): Diese Adressen schreiben Daten in die Register für die Horizontalposition der Geschosse.(Vgl. Beschreibg. von HPOSPO)

D7	D6	D5	D4	D3	D2	D1	DO

VDELAY (Vertikale Verzögerung) (DO1C): Diese Adresse schreibt Daten in das Register für Vertikalverzögerung.



D7	D6	D5	D4	D3	D2	D1	DO
P3	P2	P1	PO	M3	M2	M1	MO

VDELAY wird eingesetzt, um einzeilige Auflösung bei der vertikalen Ausrichtung eines Objekts zu bewirken, wenn zweizeilige Auflösung programmiert ist. Durch eine 1 in DVDELAY wird das entsprechende Objekt um eine Bildzeile nach unten gerückt.

Wenn die DMA für Spieler/Geschoss eingeschaltet ist, dann wird die Änderung der Vertikallage eines Objekts dadurch erreicht, dass Bits im Speicherplan verschoben werden. Ist die DMA abgeschaltet, so kann die Vertikallage per Assemblercode beeinflusst werden. Dieser lädt Daten an der gewünschten Zeile in die Graphikregister.

MOPF, M1PF, M2PF, M3PF (Geschosstreffer auf Spielfeld)(D000, D001, D002, D003): Diese Adressen lesen Geschosstreffer auf dem Spielfeld. Ein 1-Bit bedeutet, dass seit dem letzten HITCLR ein Treffer entdeckt worden ist.

nicht benutzt (Null erzwungen)	D3	D2	D1	DO	
	3	2	1	0	Spielfeldart.

POPF, P1PF, P2PF, P3PF (Spielertreffer auf Spielfeld)(D004, D005, D006, D007): Diese Adressen lesen Treffer von Spielern auf das Spielfeld.

nicht benutzt (Null erzwungen)	D3	D2	D1	DO	
	3	2	1	0	Spielfeldart.

MOPL, M1PL, M2PL, M3PL (Geschosstreffer auf Spieler)(D008, D009, DOOA, DOOB): Diese Adressen lesen Geschosstreffer auf Spielerobj.

nicht benutzt (Null erzwungen)	D3	D2	D1	DO	
	3	2	1	0	Spielernummer.

POPL, P1PL, P2PL, P3PL (Spielertreffer auf Spieler)(DOOC, DOOD, DOOE, DOOF): Diese Adressen lesen Treffer von Spielern auf Spieler.

nicht benutzt (Null erzwungen)	D3	D2	D1	DO	
	3	2	1	0	Spielernummer.

(Spieler 0 gegen Spieler 0 ergibt immer 0, ebenso Sp.1 gegen Sp.1 usw.)

HITCLR (Kollision ("Hit")gelöscht)(D01E): Diese schreibende Adresse löscht alle oben beschriebenen Kollisionsbits.

nicht benutzt

F. AUDIO

AUDCTL (Audiosteuerung)(D208): Diese Adresse schreibt Daten in die Kontrollregister für den Audioeingang (bgl. Zweitonbit 3 bei SKCTL und Anmerkungen dazu)

D7	D6	D5	D4	D3	D2	D1	D0

D7 : Ändere 17-Bit-Mehrfachschaltung in 9-Bit unterhalb Mehrfach,
D6 : Takte Kanal 1 mit 1,79 Mhz statt mit 64 Khz,
D5 : " " 3 " " " " " " ",
D4 : " " 2 " " " " " " ",
D3 : " " 4 " " " " " " ",
D2 : Schiebe Hochtonfilter in Kanal 1 ein, getaktet d. Kan. 3,*
D1 : " " " " " 2 " 2 2 2 4,
D0 : Ändere die normale Frequenz (64 Khz) in 15 Khz ab.

Exakte Frequenzen: Die oben genannten Frequenzen sind Näherungswerte. Die exakte jeweilige Frequenz, mit der die N-Teiler getaktet werden (fin) ist aus Spezialunterlagen zu ermitteln.

Die Normalformel zur Ermittlung der Output-Frequenz lautet:

$$F_{out} = F_{in}/2N$$

Dabei ist N = der Binärzahl im Frequenzregister (AUDF) plus 1 (N=AUDF + 1). Eine modifizierte Formel muss angewandt werden, wenn Fin = 1,79 Mhz beträgt und ein genaueres Ergebnis gewünscht wird:

$$F_{out} = \frac{F_{in}}{2(AUDF + M)}$$

Dabei ist: M = 4, wenn ein 8-Bit-Zähler vorliegt (AUDCTL-Bit 3 oder 4 = 0)
M = 7, wenn ein 16-Bit-Zähler vorliegt (AUDCTL-Bit 3 oder 4 = 1)

---

* vgl. Abschnitt II

AUDF1, AUDF2, AUDF3, AUDF4 (Audiofrequenz)(D200, D202, D204, D206)
Diese Adressen schreiben Daten in jedes der vier Kontrollregister für die Audiofrequenz. Jedes Register steuert einen N-Teiler.

D7	D6	D5	D4	D3	D2	D1	D0	"N"
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	2
			ETC.					
1	1	1	1	1	1	1	1	256

N.B.: "N" ist um Eins größer als die binäre Zahl im Audio-Frequenzregister AUDF(X)

AUDC1, AUDC2, AUDC3, AUDC4 (Audiokanal-Steuerung)(D201, D203, D205, D207): Diese Adressen schreiben Daten in jedes der vier Audio-Kontrollregister. Jedes dieser Register steuert den Geräuschinhalt und die Lautstärke des zugehörigen Audiokanals.

	Geräuschinhalt bzw. Verzerrung				Lautstärke				
HEX	D7	D6	D5	D4	D3	D2	D1	D0	Der vom Frequenzregister gesetzte "N"-Teiler.
0	0	0	0	0					- 17 Bit mehrfach - 5 Bit mehrfach - N
2	0	0	1	0					- 5 Bit mehrfach - N - 2
4	0	1	0	0					- 4 Bit mehrfach - 5 Bit mehrfach - N
6	0	1	1	0					- 5 Bit mehrfach - N - 2
8	1	0	0	0					- 17 Bit mehrfach - N
A	1	X	1	0					- reiner Ton - N - 2
C	1	1	0	0					- 4 Bit mehrfach - N
1	X	X	X	1					- forcierter Output (nur Lautstärke)
0					0	0	0	0	- niedrigste Lautstärke
8					1	0	0	0	- halbe Lautstärke
F					1	1	1	1	- höchste Lautstärke

Tonhöhenwerte für die normalen Musiknoten, - AUDCTL=0, AUDC= hex.AX.

		AUDF Hexadez.	AUDF Dezimal
Hohe Noten	C	1D	29
	B	1F	31
	A# od. Bb	21	33
	A	23	35
	G# od. Ab	25	37
	G	28	40
	F# od. Gb	2A	42
	F	2D	45
	E	2F	47
	D# od. Eb	32	50
	D	35	53
	C# od. Db	39	57
	C	3C	60
	B	40	64
	A# od. Bb	44	68
	A	48	72
	G# od. Ab	4C	76
	G	51	81
	F# od. Gb	55	85
	F	5B	91
	E	60	96
	D# od. Eb	66	102
	D	6C	108
	C# od. Db	72	114
Mittleres C	C	79	121
	B	80	128
	A# od. Bb	88	136
	A	90	144
	G# od. Ab	99	153
	G	A2	162
	F# od. Gb	AD	173
	F	B6	182
	E	C1	193
	D# od. Eb	CC	204
	D	D9	217
Tiefe Noten	C# od. Db	E6	230
	C	F3	243

STIMER (Start Timer)(D209): **Diese (schreibende) Adresse stellt alle Audio-Frequenz-Teiler auf ihren "AUDF"-Wert zurück. Diese Teiler erzeugen Timer-Unterbrechungen, wenn sie auf Null zurückzählen (falls sie per IRQEN eingeschaltet sind)** (Vgl. auch IRQST).

nicht benutzt

RANDOM (Zufallszahlen-Generator) (D20A): **Diese Adresse liest die 8 oberen Bits eines 17-Bit-Mehrfachzählers (bzw. 9 Bits, wenn Bit 7 des AUDCTL = 1).**

D7	D6	D5	D4	D3	D2	D1	D0

G. TASTATUR UND LAUTSPRECHER

CONSOL (Eingang Konsolenschalter)(D01F):
Diese Adresse liest oder schreibt Daten, die sich auf die Schalter und Anzeiger der Konsole beziehen (wird durch den OS-Code für Vertikal-Leertakt auf 8 gesetzt).

nicht benutzt	D3	D2	D1	D0

Vor dem Ablesen der Schalter sollte 08 in diese Adresse geschrieben werden ! - Eingeschriebene Einsen stellen die Schalterleitung ab.

Zuordnung der CONSOL - Bits:

D0 Spielbeginn
D1 Spielauswahl
D2 Möglichkeitswahl
} - 0 bedeutet, dass Knopf gedrückt,

D1 Lautsprecher - sollte auf 1 bleiben, nur momentweise auf 0. Das OS schreibt eine 1 ein, während der Vertikal-Leertakt läuft.

KBCODE (Tastaturcode)(D209): Diese Adresse liest den Tastaturcode. Sie wird gewöhnlich infolge einer tastaturunabhängigen Unterbrechung abgefragt (IRQ und Bits 6 und 7 von IRQST). Vgl.IRQEN wegen Einschaltung von tastenabhängiger Unterbrechung; vgl.SKCTL Bits 1 und 0 wegen Einschaltung der Halte- und der Absprungleitung für d.Tastatur.

D7	D6	D5	D4	D3	D2	D1	D0

D7 = Steuertaste, D6 = Transporttaste.

Lesung durch OS in das Schattenregister CH, sobald eine Taste bedient wurde. Das OS hat eine Zeichen-Abruffunktion, durch welche der Tastaturcode in ATASCII (ATARI-ASCII) verwandelt wird.

UMWANDLUNG VON TASTATURCODE IN ATASCII

Tasten-Code	Tasten-Kappe	Kl.-Buchstaben	Gross-Buchst.	Steuer-Code		Tasten-Code	Tasten-Kappe	Klein-Buchst.	Gross-Buchst.	Steuer-Code
00	L	6C	4C	0C		20	,	2C	5B	00
01	J	6A	4A	0A		21	SPACE	20	20	20
02	;	3B	3A	7B		22	.	2E	5D	60
03						23	N	6E	4E	0E
04						24				
05	K	6B	4B	0B		25	M	6D	4D	0D
06	+	2B	5C	1E		26	/	2F	3F	
07	*	2A	5E	1F		27	[Atari-Logo]	*	*	*
08	O	6F	4F	0F		28	R	72	52	12
09						29				
0A	P	70	50	10		2A	E	65	45	05
0B	U	75	55	15		2B	Y	79	59	19
0C	RET	9B	9B	9B		2C	TAB	7F	9F	9E
0D	I	69	49	09		2D	T	74	54	14
0E	-	2D	5F	1C		2E	W	77	57	17
0F	=	3D	7C	1D		2F	Q	71	51	11
10	V	76	56	16		30	9	39	28	
11						31				
12	C	63	43	03		32	0	30	29	
13						33	7	37	27	
14						34	BACKS	7E	9C	FE
15	B	62	42	02		35	8	38	40	
16	X	78	58	18		36	<	3C	7D	7D
17	Z	7A	5A	1A		37	>	3E	9D	FF
18	4	34	24			38	F	66	46	06
19						39	H	68	48	08
1A	3	33	23	*		3A	D	64	44	04
1B	6	36	26			3B				
1C	ESC	1B	1B	1B		3C	CAPS	*	*	*
1D	5	35	25			3D	G	67	47	07
1E	2	32	22	FD		3E	S	73	53	13
1F	1	31	21	*		3F	A	61	41	01

* Spezialfälle

H. AUSGANG FÜR SERIELLE ÜBERTRAGUNG. (Vgl.Konsolenstecker)

SKCTL (Kontrolle serielle Aussenübertragung)(D2OF): Diese Adresse schreibt Daten in das Kontrollregister für die Schaltkombination der Aussenübertragung, ausserdem für die Paddlepot-Schnellschaltung und die Tastatur-Eingabe.

Die Bits stehen normalerweise auf Null; auf 1 erfüllen sie die unten angegebenen Funktionen:

D7	D6	D5	D4	D3	D2	D1	DO

D7 Erzwungener Abbruch (erzwingt 0 i.d. seriellen Ausgabe (Zwischenraum)) +

D6, D5, D4 } Gangkontrolle für serielle Übertragung (vgl.Gangübersicht am Ende der Beschreibung der seriellen Übertragung, vor.Kapitel)

D3 zweitonig (serieller Output, übertragen als zweitoniges Signal anstatt als logisch wahr bzw. logisch falsch),

D2 Schnellgang für Paddlepot-Ablesung (der Zeilenzähler des Paddlereglers läuft in der Zeit von nur zwei Bildzeilen statt in der Zeit eines ganzen Bilddurchlaufs ab. Die Entlade-Transistoren für den Kondensator werden völlig abgeschaltet).

D1 Anschaltung der Tasten-Halteleitung,

DO-D1 (beide 0) Einrichten des Systems (Zustand, der zum Testen und Vorbereiten des Chips benötigt wird). ++

OS-Schattenregister: SSKCTL (hexad. 232).

Das OS schaltet die Halte- und Absprungleitungen für Tasten an und ändert unter Umst. die anderen Bits für verschiedene I/O-Operationen. Speziell eine abgebrochene Kassettenoperation kann das Zweitonbit auf 1 stehen lassen, so dass unschöne Audiosignale entstehen. Dies lässt sich durch Einschreiben von hexad. 13 sowohl in das SKCTL wie in das SSKCTL nach dem I/O bzw. vor dem Modifizieren der Audioregister korrigieren.

+ Nach Einschalten ist der Stecker für serielle Übertragung unter Umständen ohne Strom. Um die Übertragung zu aktivieren, sende man ein Byte nach aussen (am besten 00 oder FF).

++ es gibt keinen ursprünglichen Zustand "eingeschaltet". POKEY hat keinen Rückstellstift.

SKSTAT (Serialübertragungs- bzw. Tastatur-Zustand)(D20F): Diese Adresse liest das Statusregister (Zustandsregister), das über die serielle Übertragung bzw. die Tastatur informiert.

D7	D6	D5	D4	D3	D2	D1	D0

Die Bits stehen normalerweise auf wahr. Auf Null ergeben sie folgende Informationen:

D7 = 0 = Fehler im seriellen Eingabe-Rahmen,
D6 = 0 = serielle Eingabe "überlaufen",
D5 = 0 = Tastatureingabe "überlaufen",
D4 = 0 = direkt vom Eingang f. serielle Eingabe,
D3 = 0 = Transporttaste gedrückt,
D2 = 0 = Letzte Taste noch gedrückt,
D1 = 0 = Verschiebereg. f. ser.Eingabe arbeitet noch
D0 = 1 = nicht benutzt (log."wahr")

Sperren müssen auf 1 zurückgestellt werden (SKRES) -

D5 u. D6 werden auf Null gesetzt, wenn neue Daten und dasselbe Bit des IRQST Null ist.

SKRES (Zurückstellen des oben genannten Statusregisters)(D20A): Diese (schreibende) Adresse setzt Bits 7, 6 u.5 des Statusregisters für die serielle Übertragung auf 1 zurück.

nicht benutzt

SERIN (serielle Eingabedaten)(D20D): Diese Adresse liest das 8-Bit-Parallel-Halteregister, das geladen wird, sobald ein vollständiges Byte von Eingabedaten empfangen wurde. Diese Adresse wird gewöhnlich in Abhängigkeit von einer seriellen Information im Unterbrechungstakt gelesen (IRQ und Bit 5 von IRQST), vgl. auch IRQEN.

D7	D6	D5	D4	D3	D2	D1	D0

Kontaktstifte des Steckers für serielle Eingabe/Ausgabe:



1. Eingabetakt,
2. Ausgabetakt,
3. Dateneingabe im Computer,
4. GND,
5. Datenausgabe aus Computer,
6. GND,
7. Befehl,
8. Motorkontrolle,
9. Weiter,
10. +5
11. Audio-Eingabe,
12. +12,
13. Unterbrechung.

vgl. die Beschreibung der seriellen Aussenübertragung im OS-Handbuch wegen Details.

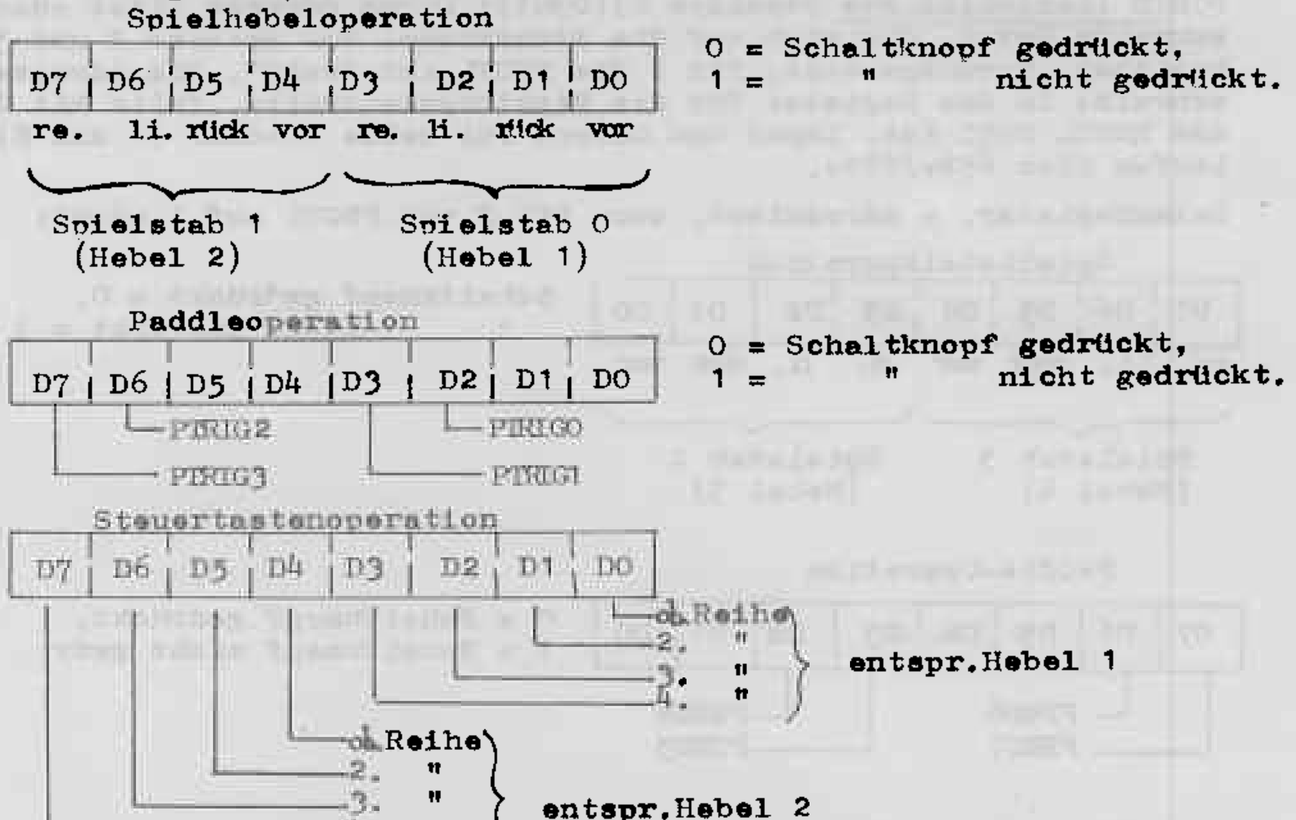
SEROUT (serielle Ausgabedaten)(D20D): Diese Adresse schreibt in das 8-Bit-Parallelhalteregister, dessen Inhalt in das serielle Verschieberegister für die Ausgabe übertragen wird, wenn ein vollständiges Ausgabedatenbyte gesendet wurde. Sie wird gew. abhängig von einer Datenausgabe-Unterbrechung benutzt (IRQ u. Bit 4 des IRQST).

D7	D6	D5	D4	D3	D2	D1	DO

I. STECKER FÜR AUSSENSTEUERUNGEN (Vorderseite der Konsole).

PORTA (Steckverbindg.A)(D300): Diese Adresse liest oder schreibt Daten, die sich auf Spieler 0 und Spieler 1 beziehen, wenn Bit 2 des PACTL auf "wahr" steht (1). Sie schreibt an das Richtungskontrollregister, wenn Bit 2 des PACTL auf 0 steht. Input/Output für beide Stecker (A und B) läuft über 652o/682o.

Datenregister, adressiert bei Bit 2 des PACTL = 1:



Register für Richtungskontrolle, - adressiert, wenn Bit 2 des PBCTL = 0.

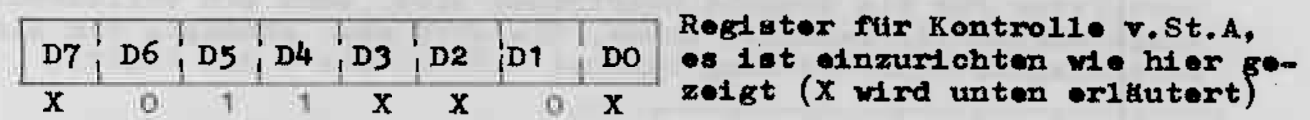
D7	D6	D5	D4	D3	D2	D1	DO

Jedes Bit entspricht einem Hebelkontakt.

0 = Eingabe,
1 = Ausgabe.

OS-Schattenregister: STICKO (hexadez. 278), STICK1 (279), PTRIGO-3 (27C - 27F).

PACTL (Kontrolle Stecker A)(D302): Diese Adresse schreibt oder liest Daten in das bzw. von dem Register für Stecker A.



Register für Kontrolle v.St.A, es ist einzurichten wie hier gezeigt (X wird unten erläutert)

D7 - (nur lesend) Statusbit für Unterbrechung bei Peripherieger. A, "weiter"-Leitung für Übertragungskanal. Wird zurückgeschaltet durch Lesen des Registers für Stecker A. Wird umgeschaltet durch Unterbrechung bei Peripheriegerät A.

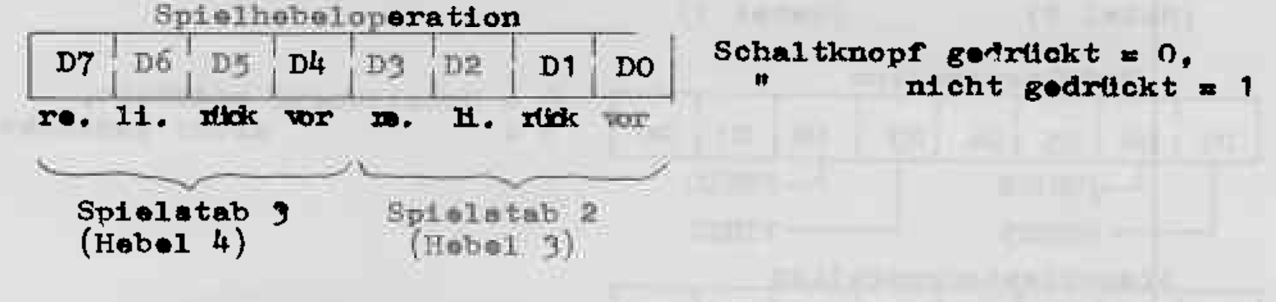
D3 - Motorkontrolle f. Peripheriegerät über Schreibkanal (0 = an, 1 = aus.)

D2 - Kontrolliert Stecker A, - Adressierung wie oben beschrieben (schreibend)(1= Reg.f.Stecker A, 0=Richtungskontr.,

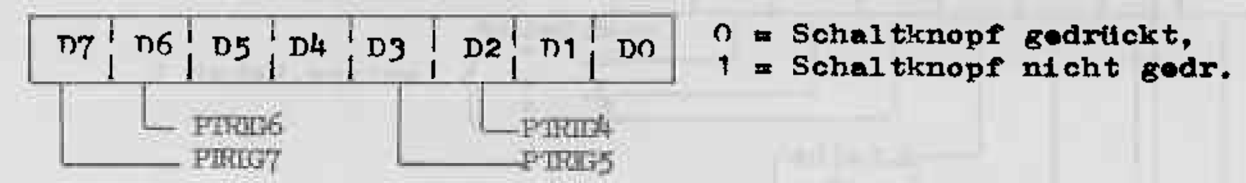
D0 - Einschaltbit für die Unterbrechung von Peripheriegerät A, (schreibend)(1 = eingeschaltet). Rückstellung durch Hauptschalter "ein" oder durch Prozessor. Setzung durch Prozessor.

PORTB (Kontrolle des Steckers B)(D301): Diese Adresse liest oder schreibt Daten, die sich auf die Steuerhebel von Spieler 2 und 3 beziehen, vorausgesetzt, Bit 2 des PBCTL ist "wahr". Die Adresse schreibt in das Register für die Richtungskontrolle, falls Bit 2 des PBCTL Null ist. Input und Output für beide Stecker (A und B) laufen über 652o/682o.

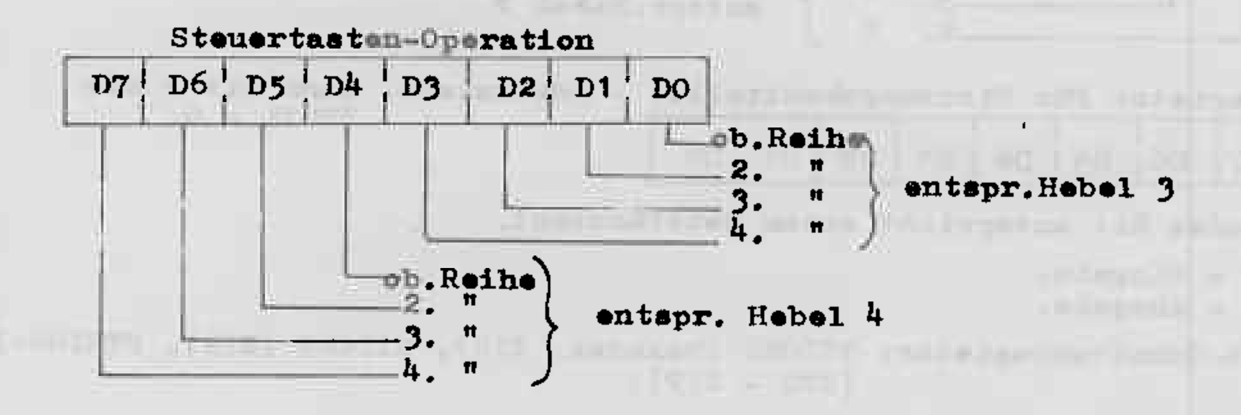
Datenregister, - adressiert, wenn Bit 2 von PBCTL auf 1 steht:



Schaltknopf gedrückt = 0,
" nicht gedrückt = 1



0 = Schaltknopf gedrückt,
1 = Schaltknopf nicht gedr.



Register für Richtungskontrolle, - adressiert, wenn Bit 2 des PBCTL=0.

D7	D6	D5	D4	D3	D2	D1	D0

Jedes Bit entspricht einem Spielhebel-Kontakt.
0 = Eingabe,
1 = Ausgabe.

OS-Schattenregister: STICK2 (hexadez. 27A), STICK3 (27B), PTRIG4-7 (280-283).

PBCTL (Kontrolle Stecker B)(D303): Diese Adresse schreibt oder liest Daten in das bzw. aus dem Kontrollregister für Stecker B.

Nur lesend

D7	D6	D5	D4	D3	D2	D1	D0
X	0	1	1	X	X	0	X

Kontrollregister Stecker B. Es ist einzurichten wie hier gezeigt (X wird weiter unten beschrieben).

D7 - (nur lesend)Statusbit für Unterbrechung von Peripheriegerät B. Unterbrechungsleitung für seriellen Kanal. Wird rückgestellt durch Lesen des Registers für Stecker B, wird gesetzt durch Unterbrechung bei Peripheriegerät B.

D3 - Erkennung von Peripheriebefehlen. Befehlsleitung zur seriellen Übertragung.

D2 - Kontrolle der Adressierung von Stecker B wie oben beschrieben. (1 = Register für Stecker B, 0 = Register f. Richtungskontrolle).

D0 - Einschaltbit für Unterbrechung von Peripheriegerät B. 1 = Ein. Rückstellung durch Hauptschalter "ein" oder durch Prozessor. Setzung durch Prozessor. Wird auf hexadez. 3 C gesetzt durch den IRQ-Code des OS.

POT0 - POT7 (Paddlewerte)(D200 - D207): Diese Adressen lesen die Werte (0 - 228) von 8 Paddlesteuerungen, die mit den 8 Leitungen der Potstecker verbunden sind. Die Paddlesteuerungen werden von links nach rechts numeriert (Blickrichtung auf Konsolen-Vorderseite). Wenn man den Drehknopf des Paddles nach rechts bewegt, so schaltet man die Potwerte herunter. Die Werte gelten erst 228 TV-Zeilen nach dem Befehl POTGO (s.unten) oder nach einem sog. ALLPOT-Wechsel.

D7	D6	D5	D4	D3	D2	D1	D0

jeweiliger Potwert (0-228)

OS-Schattenregister: PADDL0 - 7 (hexadez. 270 - 277).

ALLPOT (All Pot Lines)(Alle Potleitungen gleichzeitig)(D208): Diese Adresse liest den jeweiligen Zustand des achtfachen Potsteckers ab.
Die Entladetransistoren für den Messkondensator müssen dadurch abgeschaltet werden, dass entweder die Schnellübertragung des Pot (Bit 2 des SKCTL) oder die normale Potübertragung programmiert wird (POTGO)

D7	D6	D5	D4	D3	D2	D1	DO

Potnummer:
7 6 5 4 3 2 1 0
(8 Zustände der Potleitungen)

0 = Wert des Potregisters gilt,
1 = Wert des Potregisters gilt nicht.

POTGO (Start der Potübertragung)(D20B):

Keine Datenbits benutzt

Diese nur schreibende Adresse startet die Abfolge der Potübertragung. Zuerst sollten die Potwerte (POTO - POT7) gelesen werden. Dieser Schreibimpuls wird dann für die folgende Schrittfolge eingesetzt:
1. Streckenzähler (scan counter) wird auf Null gebracht,
2. Abschaltung der Entladetransistoren,
3. Streckenzähler beginnt zu zählen,
4. Zählerwert wird in jedem der 8 Register (POTO - 7) festgehalten, während jede Potleitung die Auslösespannung überschreibt.
5. Zähler erreicht 228, die Entladetransistoren schalten sich ein.

(Die Schreibung erfolgt durch den OS-Vertikal-Leer-Code)

TRIGO, TRIG1, TRIG2, TRIG3 (Auslöser-Kontakte)(0 DO10, 1 DO11, 2 DO12, 3 DO13): Diese Adressen lesen die Steckerkontaktstäbe, die normalerweise an die Spielstab-Auslöser gekoppelt sind.

nicht benutzt (Null erzwungen)'	DO

0 = Auslöseknopf gedrückt,
1 = Knopf nicht gedrückt.

OS-Schattenregister: STRIGO - 3 (hexadez. 284 - 287)

Anmerkung: TRIGO - TRIG3 werden normalerweise direkt vom Prozessor gelesen. Wenn jedoch Bit 2 des GRACTL auf 1 steht, werden diese Eingaben gesperrt, sobald sie auf logisch Null gehen. Die Sperren werden aufgehoben (1 wird geladen), wenn Bit 2 des GRACTL auf 0 gebracht wird.

PENH (horizontale Colorclock-Position d. Lichtstifts)(D40C): Diese Adresse liest das Register für die Lichtstift-Horizontale, das auf dem Colorclock-Zähler der Hardware basiert. Die Werte reichen von 0 - 227. Bildumbruch setzt ein, sobald der Stift die rechte Kante des Normalmonitors erreicht. PENH und PENV werden modifiziert, sobald eine der Leitungen für die Spiel-Auslöser abgeschaltet wird.

D7	D6	D5	D4	D3	D2	D1	D0
H7	H6	H5	H4	H3	H2	H1	H0

OS-Schattenregister: LPENH (hexadez. 234).

PENV (Vertikale Zeilenposition des Lichtstiftes)(D40D): Diese Adresse liest das Register für die Lichtstift-Vertikale (die 8 höchsten Bits, ebenso wie bei VCOUNT).

D7	D6	D5	D4	D3	D2	D1	D0
LP8 7	6	5	4	3	2	1	0

LP0 wird nicht gelesen. Die Auflösung ist zweizeilig.

OS-Schattenregister: LPENV (hexadez.235)

Fronthebel (Steuerhebel) als Input/Output - Einheiten:
PIA (652o/682o)
Output: TTL - Spannungsebenen, 1 Ladung,
Input: TTL - Spannungsebenen, 1 Ladung.

Schaltung von Stecker A (typischerweise):



Schaltung von Stecker B (typischerweise):

o+5
4,7K
652o (B)
Stecker
22o
0,001
Hebel

Schaltg.f."Auslöser"-Stecker (typischerweise):



Kontaktstifte des Steuerhebelsteckers:

Ausgänge (Konsole)	Eingänge (Steckkontakt)
1 2 3 4 5 / 6 7 8 9	5 4 3 2 1 / 9 8 7 6

Kont. Stift	Steuergeräte: Spielstab	Steuergeräte: Paddle (Pot)	Steuergeräte: Tastatur	HARDWARE-REGISTER	OS-VARIABLEN
1	vorwärts		Obere Reihe +	Bit 0 oder 4 ++	Bit 0 +++
2	rückwärts		2. Reihe +	Bit 1 od. 5 ++	Bit 1 +++
3	n. links	A (li.Auslöser)	3. Reihe +	Bit 2 od. 6 ++	PTRIG0,2,4,6
4	n. rechts	B (re.Auslöser)	unt.Reihe +	Bit 3 od. 7 ++	Bit 2 +++ PTRIG1,3,5,7
5		POT B (rechts)	1. Spalte	POT 1,3,5,7	Bit 3 +++ PADDL1,3,5,7
6	Auslöser		3. Spalte	TRIG0,1,2,3	STRIG0,1,2,3
7		+5	+5		
8	GND	GND			
9		POTA (links)	2. Spalte	POT 0,2,4,6	PADDL0,2,4,6

+ schreibend,
++ PORTA oder PORTB,
+++ Stab 0, 1, 2 oder 3.